Tutorial Talk1

[Notice] This is an edited version of the material provided by CVPR2024 Tutorial
https://multimodalagentai.github.io/
https://multimodalagentai.github.io/files/JuanCarlosNiebles_CVPR2024_Tutorial_GeneralistAgentAI.pdf

Language-based AI Agents and Large Action Models (LAMs)

CVPR 2024 Tutorial “Generalist Agent AI”

Juan Carlos Niebles, Research Director
@jcniebles | www.niebles.net
He/Him



LLM based AI Agents

Juan Carlos Niebles, Salesforce

Conversational AI Assistants

Work Agent

Hello I'm Einstein Copilot your trusted AI assistant

Can you help route my case for approval?

I have modified this case status to Escalated and created an escalation request with the subject "Modify deliver center to Portland, OR".

[Edit Case]

Personal Agent

Let's start booking my summer vacation travel. Find and book family friendly hotel for my upcoming stay in France..

I found this nice hotel near the venues you are planning to visit. I booked 2 rooms for the full duration of your stay.

[View Booking]

Creative Agent

I'm writing a fiction story about animals in the jungle. What are some adventures these animals might have? Give me some ideas and illustrate them with pictures.

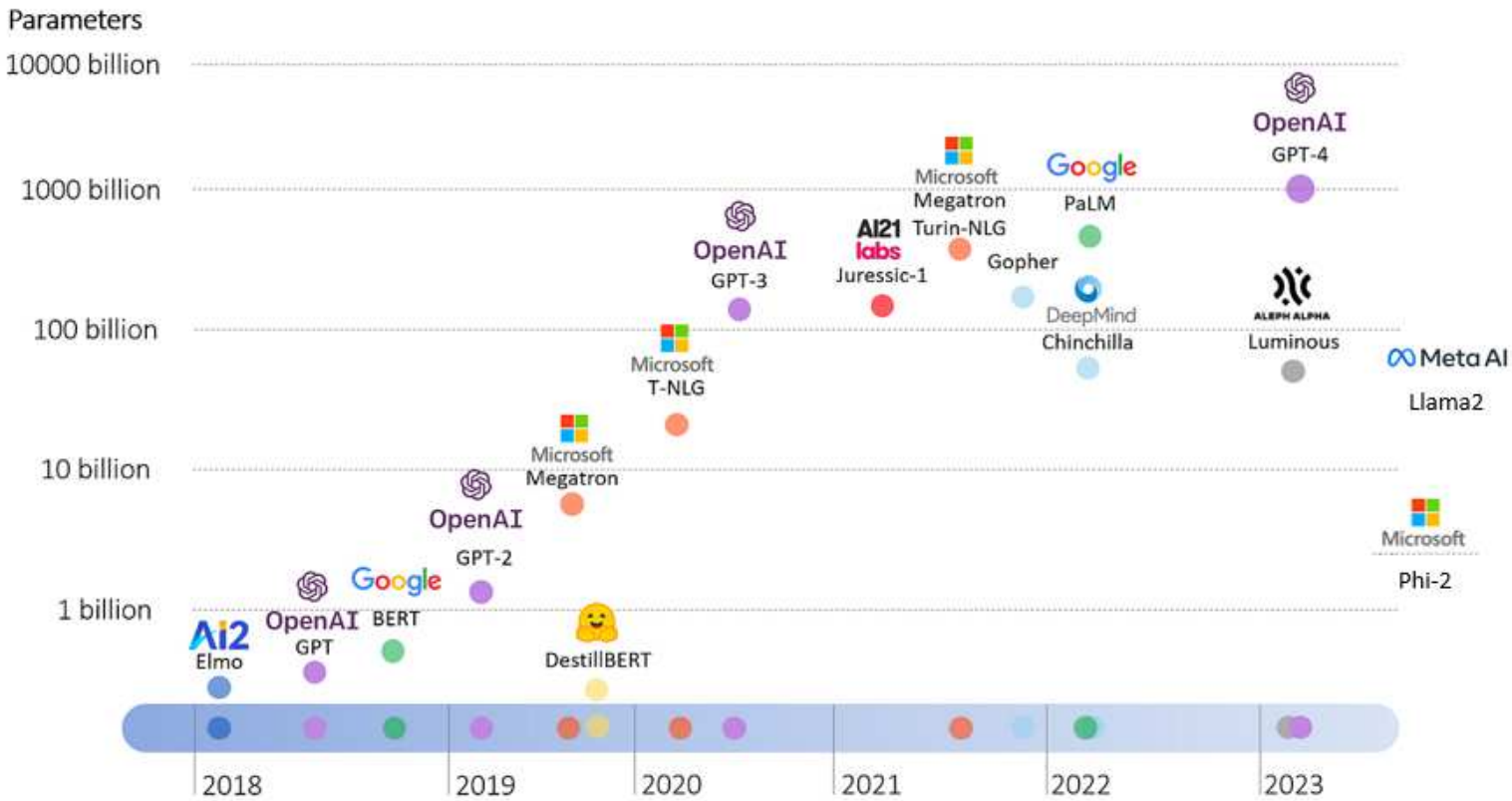
Sure, some ideas:
1. The animals might organizing a party.



2. The animals may need to solve a problem [...]

Juan Carlos Niebles, Salesforce

LLM Parameter count



Source: https://www.predli.com/post/llm-deep-dive-phi-2

LLM based AI Agents

2024~2025 SOTA LLMs

[☞ ChatGPT o1 DeepResearch]

MMLU(Massive Multitask Language Understanding) - 57개 과목

모델	기업 / 공개여부	파라미터 수	대표 벤치마크 (MMLU 등)	최대 컨텍스트	특징 & 비고
GPT-4o	OpenAI (폐쇄)	비공개	MMLU ≈ 87 % paperswithcode	128 K 토큰 momen	멀티모달(텍스트·음성·비전), 실시간 스트리밍, 고속 토큰 출력
Claude 3 Opus	Anthropic (폐쇄)	비공개	MMLU ≈ 86.8 % lifearchitect.substack	200 K (최대 1 M 연구용) anthropic	긴 문맥·안전성 강조, 다국어 정확도 90 % ↑
Gemini 1.5 Pro	Google DeepMind (폐쇄)	비공개	MMLU ≈ 85.9 % artificialanalysis	1 M 토큰 blog	가장 긴 상용 컨텍스트, 멀티모달·툴콜, 가격 지속 인하
Llama 3 70B	Meta (오픈 가중치)	70 B	MMLU ≈ 86.0 % paperswithcode	8 K 토큰 huggingface	SOTA 공개 모델, 상업·연구 모두 무료 라이선스
Llama 3 405B	Meta (사내)	405 B	MMLU ≈ 86.6 % paperswithcode	8 K	GPT-4o와 대등, 아직 공개되지 않음
Mistral Large 2	Mistral AI (연구/상업 라이선스)	123 B mistral	MMLU ≈ 84 % docsbot	128 K 토큰 docsbot	단일 노드 추론 가능, 다국어·코딩 강점
Grok-1.5	xAI (부분 공개)	314 B MoE x	GSM8K 90 %, HumanEval 74.1 % x	128 K 토큰 x	유머 지향, 수학·코딩 대폭 향상, MoE로 효율적 추론
Qwen 2 72B	Alibaba (Qwen, 오픈)	72 B huggingface	MMLU ≈ 84.2 % huggingface	128 K 토큰 artificialanalysis	다국어·코딩 우수, 라이선스 유연, RAG 친화
Command R+	Cohere (open weights 연구)	104 B huggingface	MMLU ≈ 88.2 %* anakin	128 K 토큰 stackoverflow	RAG·툴체이닝 특화, Chatbot Arena 상위권

- **GPT-4o**: 실시간 멀티모달 LLM의 기준을 세움. 파라미터·훈련 세부는 비공개지만 대부분의 범용 벤치마크에서 최고 수준.
- **Claude 3 Opus**: 긴 문맥·안전성에 초점. 복잡한 추론·코딩에서 GPT-4o와 호각세를 보이며, 언어 다양성도 강점.
- **Gemini 1.5 Pro**: 1 백만 토큰이라는 압도적 컨텍스트 창으로 대규모 문서·비디오 처리에 유리. 가격을 지속적으로 낮추며 시장 확대 중.
- **Llama 3 70B**: 공개 LLM 중 최상위 성능. 상용 사용도 무료라 연구·스타트업에 인기.
- **Mistral Large 2**: 123 B임에도 84 % MMLU 달성. "적당한 크기"로 비용-효율 균형.
- **Grok-1.5**: 314 B MoE 구조로 추론 시 25 % 가중치만 활성화. GSM8K 90 %로 수학 성능이 대폭 향상.
- **Qwen 2 72B**: 중국어 포함 다국어·코딩 벤치에서 두각. 18 T 토큰 이상으로 재훈련한 Qwen 2.5는 MMLU 86 %까지 상승 예정.
- **Command R+**: RAG·도구 사용 시나리오에 최적화. 128 K 문맥, 오픈 가중치(연구용)로 기업 내 자체 배포 가능.

LLM based AI Agents

Juan Carlos Niebles, Salesforce

Language-based Agents



Research Papers







- Huang et al, **Zero-shot Planners**, 2022
- Yao et al, **ReAct**, 2022
- Schick et al, **Toolformer**, 2023
- Shen et al, **HuggingGPT**, 2023
- Paul et al, **Refiner**, 2023
- Shinn et al, **Reflection**, 2023
- Madaan et al, **Self-refine**, 2023
- Park et al, **Generative Agents**, 2023
- …

2024년 이후 High Google Scholar Citation – SOTA Papers

- Zhiheng Xi et al, **AgentGym**: Evolving LLM-based Agents across Diverse Environments, 2024
- Kunlun Zhu et al, **MultiAgentBench**: Evaluating the Collaboration & Competition of LLM Agent, 2025
- Feng He et al, The Emerged Security and Privacy of LLM Agent, 2024
- Yu Shang, **AgentSquare**: Automatic LLM Agent Search in Modular Design Space, 2024
- Georg Wölflein et al, LLM Agents Making Agent Tools (**ToolMaker**), 2025

Survey Papers

- Xu Huang et al, **Understanding the planning of LLM agents: A survey**, 2024
- Zeyu Zhang, **A Survey on the Memory Mechanism of LLM-based Agents**, 2024
- Junyu Luo et al, **Large Language Model Agent**: A Survey on Methodology, Applications and Challenges, 2025
- Shangheng Du et al, **A Survey on the Optimization of LLM-based Agents**, 2025
- Asaf Yehudai et al, **A Survey on Evaluation of LLM-based Agents**, 2025

❖ Planning·Memory·Security·Automation·Benchmark 등 관심 하위주제별로 골라 읽으시면 최신 SOTA 흐름을 빠르게 파악할 수 있음

LLM based AI Agents

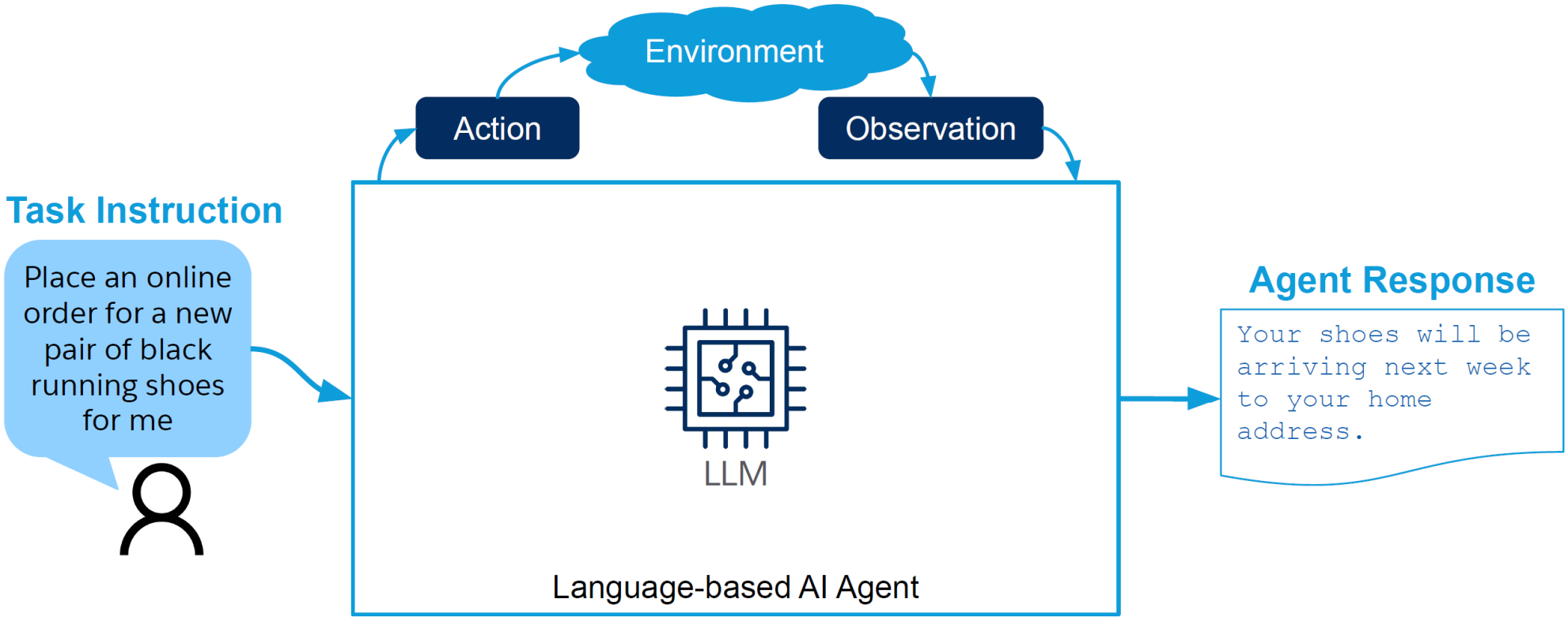
Language-based Agents

❖ (2025-05-11 기준) ⭐ 수·커뮤니티·기능 면에서 가장 활발한 "LLM-에이전트 오픈소스 프레임워크 (★: GitHub Star 수) **[☞ ChatGPT o3 LLM agent framework]**

Framework & Link		★ Stars	Licence • Main Langs	핵심 특징 / 대표 활용
LangChain (GitHub)	LangChain	≈ 107 k	MIT • Python / JS	파이프라인·툴·RAG·Agent API가 모두 포함된 **"올인원" 표준**. 가장 방대한 통합 생태계와 문서/샘플 보유.
LangGraph (GitHub)	LangChain	≈ 12 k	MIT • Python / TS	LangChain 팀이 별도로 제공하는 **그래프-형 에이전트 오케스트레이션** 레이어. 장기 메모리·체크포인트·휴먼-인-더-루프 지원.
AutoGen (GitHub)	Microsoft	≈ 44k	MIT + CC-BY • Python / .NET	다중 에이전트 채팅·팀 템플릿·GUI(Studio)·벤치마크까지 갖춘 **기업 친화적 멀티-에이전트 SDK**.
CrewAI (GitHub)	CrewAI	≈ 31k	MIT • Python	"Crew/Flow" 모델로 **역할 분담·이벤트-드리븐 제어**를 동시에 지원하는 경량 Python 프레임워크.
Semantic Kernel (GitHub)	Microsoft	≈ 24k	MIT • .NET / Python / Java	.NET·Python·Java SDK를 모두 제공, **플러그인·플래너·MCP** 내장한 마이크로서비스 급 오케스트레이터.
LlamaIndex (GitHub)	LlamaIndex	≈ 41k	MIT • Python	데이터-중심 RAG·툴 호출·멀티-에이전트 파이프라인 구현에 강력. **벡터/RDB/그래프 통합 데이터 레이어**.
Haystack v2 (GitHub)	deepset	≈ 21k	Apache-2.0 • Python	파이프라인과 Agent API를 겸비한 **LLM 오케스트레이션/RAG 엔진**. 쿠버네티스 & REST API 배포 템플릿 포함.
MetaGPT (GitHub)	FoundationAgents	≈ 55k	MIT • Python	"AI 소프트웨어 회사" 컨셉의 **역할-기반 워크플로 SOP**를 코드화. 1 줄 요구사항 → 제품 설계·코드 자동 생성.
Flowise (GitHub)	Flowise	≈ 38k	Apache-2.0 • TypeScript/React	노코드 / 로우코드 **Drag-and-Drop GUI**로 LangChain·LlamaIndex 플로우 및 Agent 조합을 시각적으로 구축.
AgentVerse (GitHub)	OpenBMB	≈ 4.5k	Apache-2.0 • Python	연구 중심 프레임워크. **시뮬레이션 vs. 태스크-솔빙** 두 모드를 제공하여 협력·경쟁 다중 에이전트 연구에 적합.

Juan Carlos Niebles, Salesforce

High-level Language-based Agent framework



Agent Benchmarking

What tasks are agents solving today?

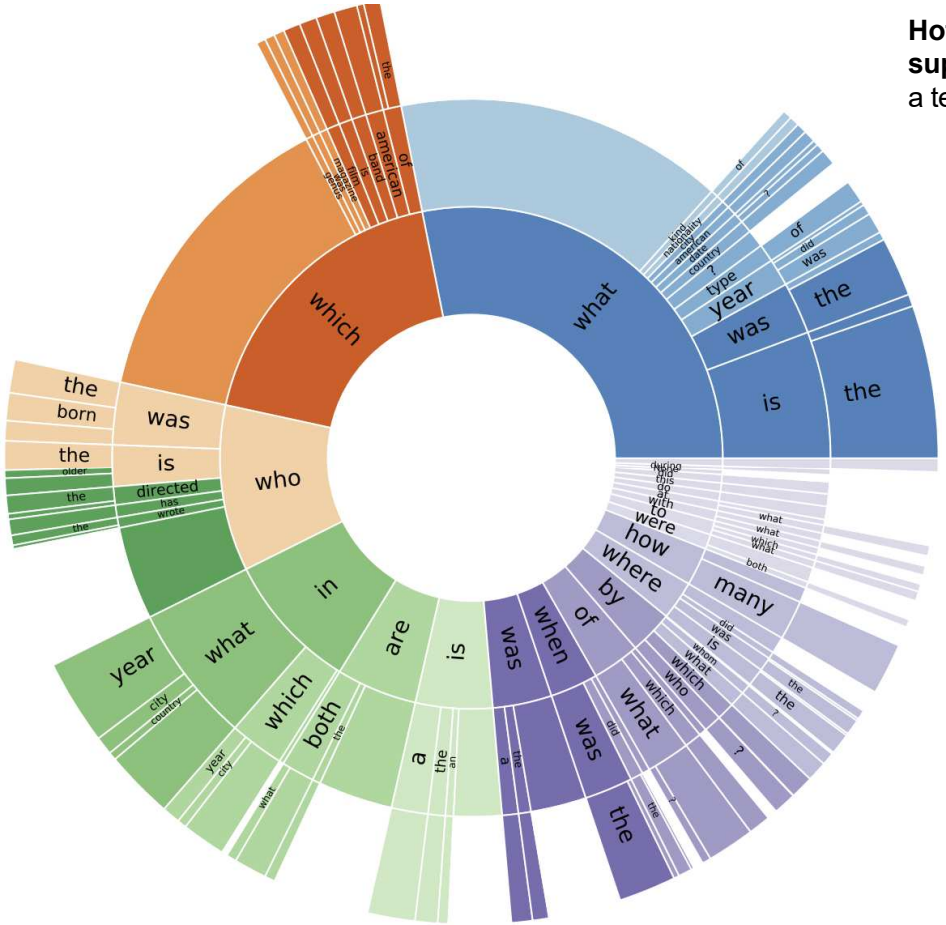


Agent Benchmarking

Juan Carlos Niebles, Salesforce

❓ HotPotQA - Knowledge Reasoning

https://hotpotqa.github.io/



HotpotQA is a **question answering dataset** featuring **natural**, **multi-hop questions**, with **strong supervision** for supporting facts to enable more explainable question answering systems. It is collected by a team of NLP researchers at Carnegie Mellon University, Stanford University, and Université de Montréal.

- 113k question + answer pairs
- Require multiple steps of reasoning and actions (Wikipedia article search, string lookup, finish)

"**User_query**":
"which magazine was started first: Arthur's Magazine or First for Women?",
"**steps**":
"**content**": "Thought: I need to Search[Arthur's Magazine] and then Search[First for Women] to collect their starting information.",
"**next observation**": "OK."

"**content**": "Action: Search[Arthur's Magazine]",
"**next observation**": "Arthur's Magazine was published in 1844."

"**content**": "Action: Search[First for Women]",
"**next observation**": "First for Women is a woman's magazine published in 1989."

"**content**": "Action: Finish[Arthur's Magazine]",
"**next observation**": "Answer is CORRECT."

[Yang et al. "**HotPotQA**: **A Dataset for Diverse, Explainable Multi-hop Question Answering**". EMNLP **2018**]

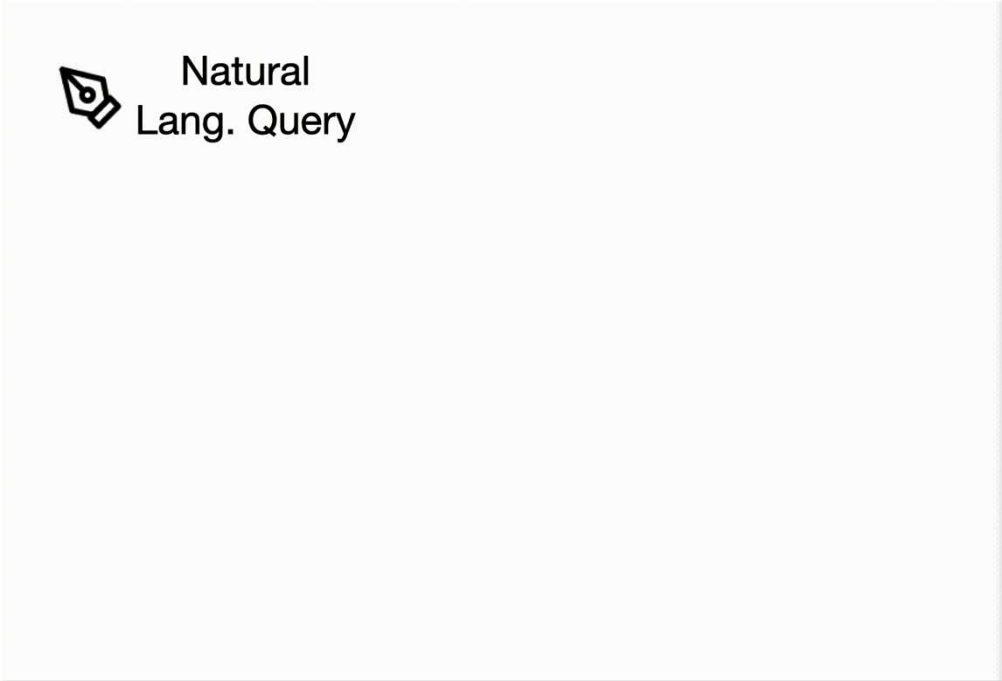
Agent Benchmarking

Juan Carlos Niebles, Salesforce

🛒 WebShop - Decision Making

httpshttps://webshop-pnlp.github.io/hotpotqa.github.io/

Towards Scalable Real-World Web Interaction with Grounded Language Agents



- Simplified e-commerce shopping environment containing 1.18M real-world products
- 12k human instructions, ground truth items
- Agent receives single instruction from user, then performs multiple actions to achieve the goal

[Yao et al “WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents”. NeurIPS **2022**]

Juan Carlos Niebles, Salesforce

🛒 WebShop - Decision Making

httpshttps://webshop-pnlp.github.io/hotpotqa.github.io/

- Towards Scalable Real-World Web Interaction with Grounded Language Agents



- Simplified e-commerce shopping environment containing 1.18M real-world products
- 12k human instructions, ground truth items
- Agent receives single instruction from user, then performs multiple actions to achieve the goal
- Interaction is purely text-based

[Yao et al “WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents”. NeurIPS **2022**]

Agent Benchmarking

🛠 ToolBench – Tool-use capability

https://github.com/OpenBMB/ToolBench

➢ To construct open-source, large-scale, high-quality **instruction tuning SFT(Supervised Fine Tuning) data**

- LLM limitations : **Limited in tool-use capabilities**, i.e., using external tools (APIs) to fulfill human instructions

→ Reason : Current instruction tuning largely focuses on basic language tasks but ignores the tool-use domain

- Related works limitations

- **Limited APIs** : Small scope of APIs with poor diversity
- **Constrained Scenario** : Confined to instructions that only involve one single tool
- **Inferior planning and reasoning** : Adapt **Chain of Thought(CoT)** or **Reasoning and Acting(ReAct)** → Cannot fully elicit the capabilities stored in LLMs and thus fail to handle complex instructions; **Error propagation** & **Limited exploration**

➢ **ToolLLM**, a general tool-use framework encompassing **data construction**, **model training**, and **evaluation**.

➢ **ToolBench**, an instruction-tuning dataset for tool use, which is constructed automatically using ChatGPT.

➢ **ToolEval;** Automatic Evaluator, Evaluate the tool-use capabilities of LLMs. Fine-tune LLaMA to obtain an LLM **ToolLLaMA**, and equip it with a neural **API retriever** to recommend appropriate APIs for each instruction.

Resource	ToolBench (this work)	APIBench (Patil et al., 2023)	API-Bank (Li et al., 2023a)	ToolAlpaca (Tang et al., 2023)	T-Bench (Xu et al., 2023b)
Real-world API?	✓	✗	✓	✗	✓
Real API Response?	✓	✗	✓	✗	✓
Multi-tool Scenario?	✓	✗	✗	✗	✗
API Retrieval?	✓	✓	✗	✗	✗
Multi-step Reasoning?	✓	✗	✓	✓	✓
Number of tools	**3451**	3	53	400	8
Number of APIs	**16464**	1645	53	400	232
Number of Instances	12657	**17002**	274	3938	2746
Number of Real API Calls	**37204**	0	568	0	0
Avg. Reasoning Traces	4.1	1.0	2.1	1.0	**5.9**

[Qin et al “ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs”, **2023**]

Agent Benchmarking

🛠 ToolBench – Tool-use capability

https://github.com/OpenBMB/ToolBench

- **ToolBench**

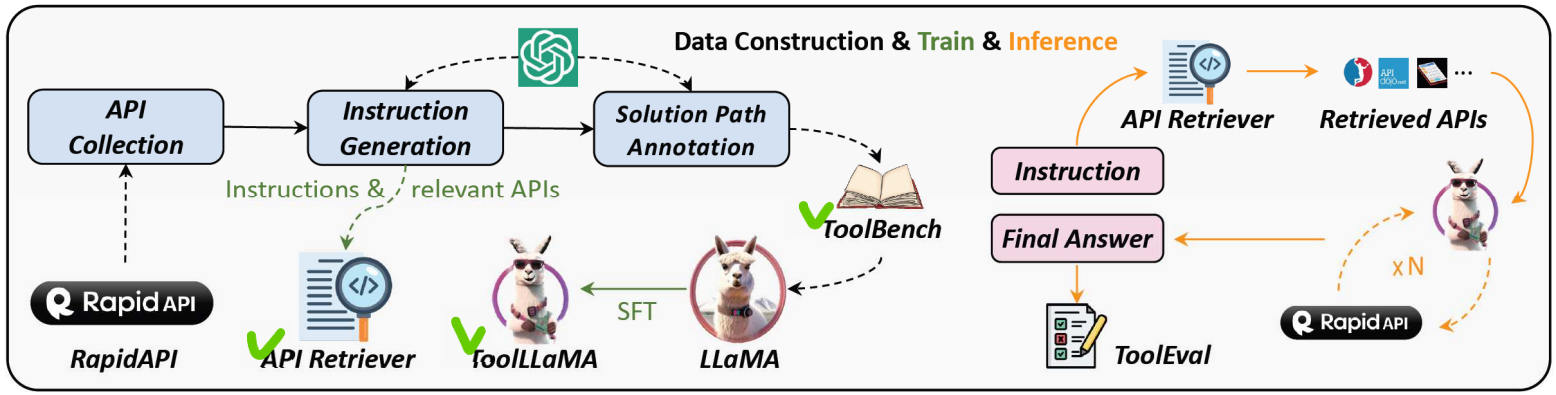


Figure 1: 3 phases of constructing **ToolBench** and how we train our **API retriever** and **ToolLLaMA**. During inference of an instruction, the API retriever recommends relevant APIs to ToolLLaMA, which performs multiple rounds of API calls to derive the final answer. The whole reasoning process is evaluated by **ToolEval**.

1) **API Collection**: Gather 16,464 REST (Representational State Transfer) APIs from **RapidAPI**, a platform that hosts massive real-world APIs provided by developers.
2) **Instruction Generation**: Curate instructions that involve both *single-tool* and *multi-tool* scenarios.

- All the data is automatically generated by OpenAI API and filtered by us, the whole data creation process is easy to scale up.

3) **Answer Annotation**: Develop a novel **depth-first search based decision tree** (**DFSDT**) to bolster the planning and reasoning ability of LLMs, which significantly improves the annotation efficiency and successfully annotates those complex instructions that cannot be answered with CoT or ReACT. We provide responses that not only include the final answer but also incorporate the model's reasoning process, tool execution, and tool execution results.
4) **API Retriever**: Incorporate **API retrieval** to equip **ToolLLaMA** with open-domain tool-using abilities.

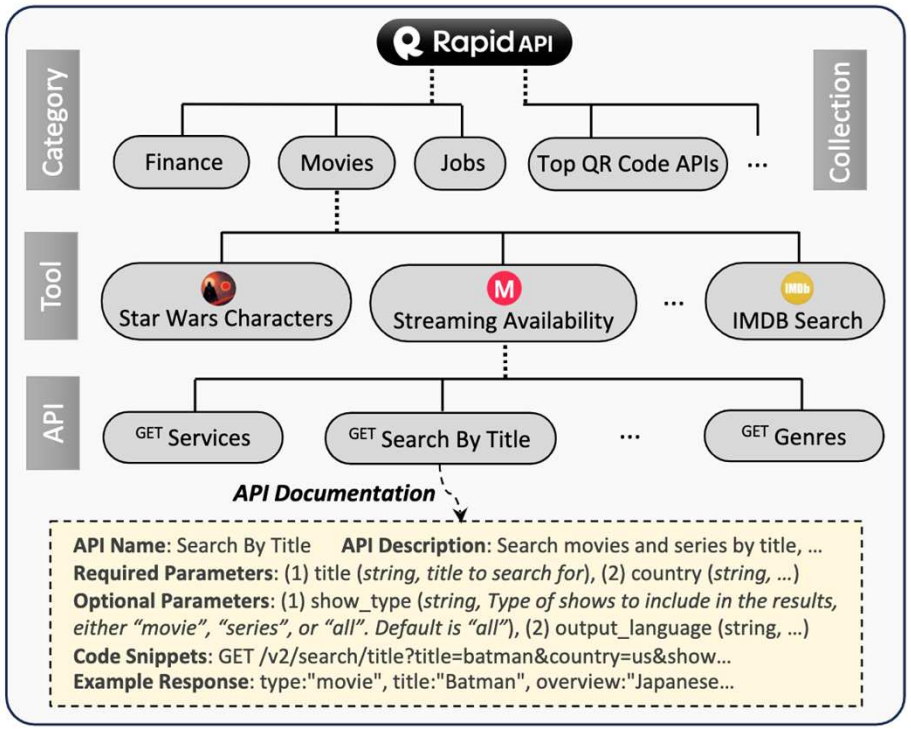
[Qin et al "ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs", **2023**]

Agent Benchmarking

🛠 ToolBench – Tool-use capability

https://github.com/OpenBMB/ToolBench

- **RapidAPI**



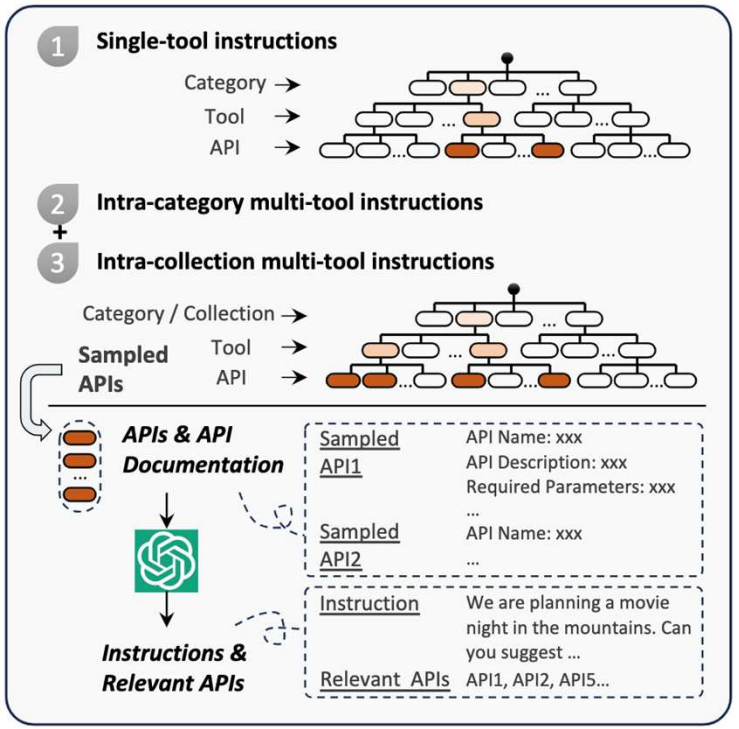


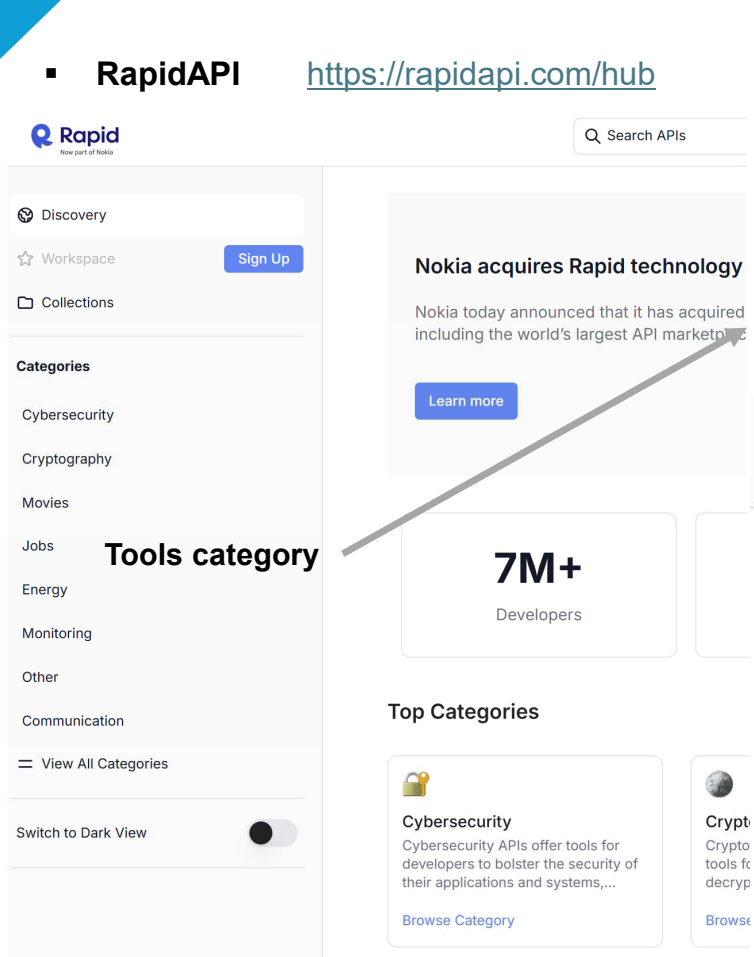
Figure 3: The hierarchy of RapidAPI (left) and the process of instruction generation (right).

We crawl **16000+** real-world APIs from RapidAPI, and curate realistic human instructions that involve them. Below we present a hierarchy of RapidAPI and our instruction generation process.

[Qin et al "ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs", **2023**]

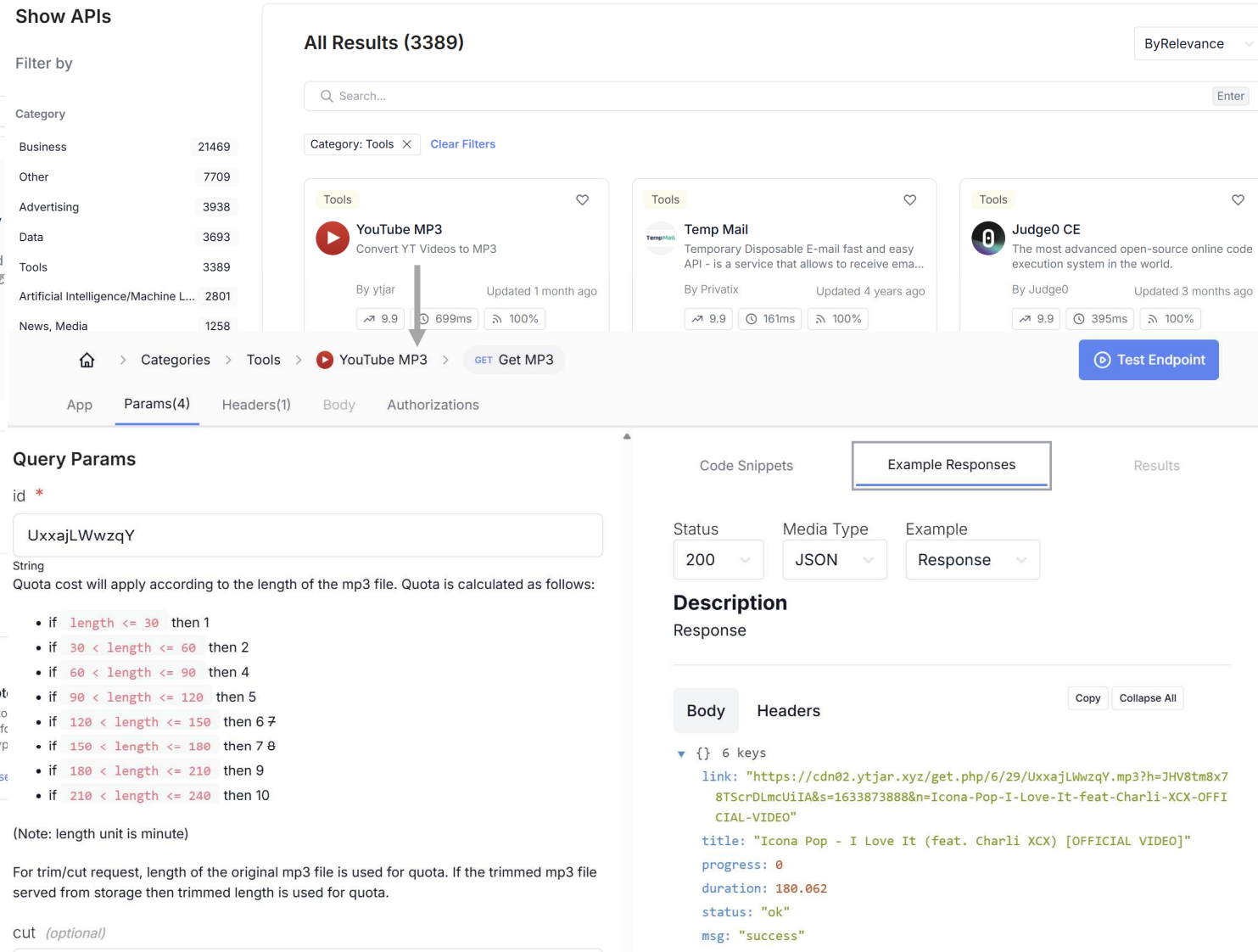
Agent Benchmarking

- **RapidAPI** https://rapidapi.com/hub



Tools category

Total 49 Categories



Juan Carlos Niebles, Salesforce

🛠 ToolBench – Tool-use capability

https://github.com/OpenBMB/ToolBench

- **DFSDT**

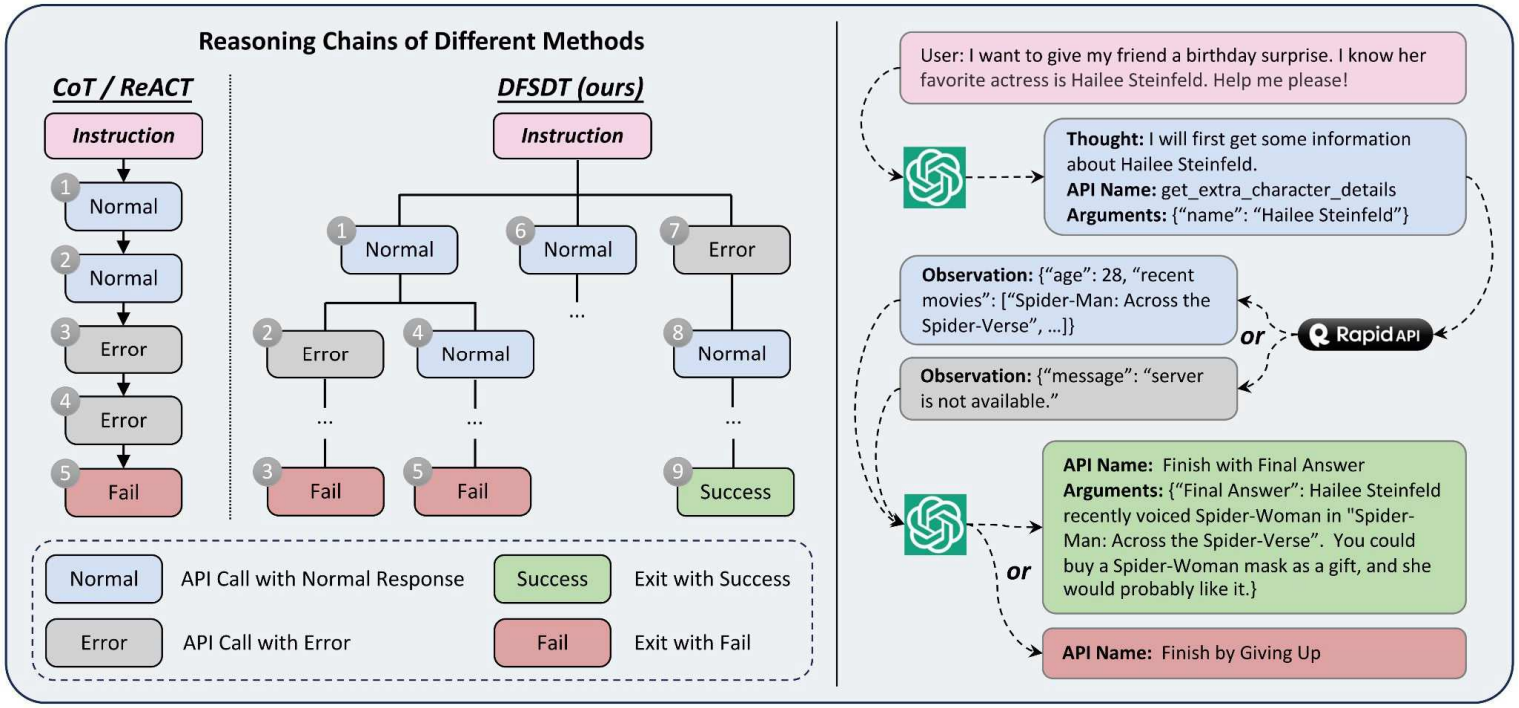


Figure 4. A comparison of our **DFSDT** and conventional **CoT** or **ReACT** during model reasoning (left). We show part of the solution path annotation process using ChatGPT (right).

[Qin et al "ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs", 2023]

- **CoT** & **ReAct** Limitations

✓ **Error propagation** : a mistaken action may propagate the errors further and cause the model to be trapped in a faulty loop, such as continually calling an API in a wrong way or hallucinating APIs.

✓ **Limited exploration** : CoT or ReACT only explores one possible direction, leading to limited exploration of the whole action space. Hence even GPT-4 often fails to find a valid solution path, making annotation difficult.

- Proposed **DFSDT** : construct a decision tree to expand the search space and increase the possibility of finding a valid path.

✓ Assess different reasoning paths and choose to either (1) proceed along a promising path or (2) abandon an existing node by calling the "Finish by Giving **Up**" function and expand a new node.

✓ During node expansion, to diversify the child nodes and expand the search space, we prompt ChatGPT with the information of the previously generated nodes and explicitly encourage the model to generate a distinct node.

✓ For the searching process, we prefer depth-first search (DFS) instead of breadth-first search (BFS) because the annotation can be finished as long as one valid path is found. Using BFS will cost excessive OpenAI API calls.

Agent Benchmarking

Juan Carlos Niebles, Salesforce

AgentBoard: evaluation framework

https://github.com/hkust-nlp/AgentBoard

➢ An Analytical Evaluation Board of Multi-turn LLM Agents

- **AgentBoard** emphasizes analytical evaluation for Large Language Models (LLMs) as generalist agents to perceive and act within various environments.
- 4 principles for constructing a benchmark to evaluate LLMs as generalist agents:

1) **Task Diversity (9tasks)**
2) **Multi-round Interaction**
3) **Partially-Observable Environments**
4) **Analytical Evaluation**

Fine-grained progress rates, grounding accuracy, performance breakdown for hard and easy examples, long-range interactions, detailed performance across various sub-skills, and trajectory with friendly visualization

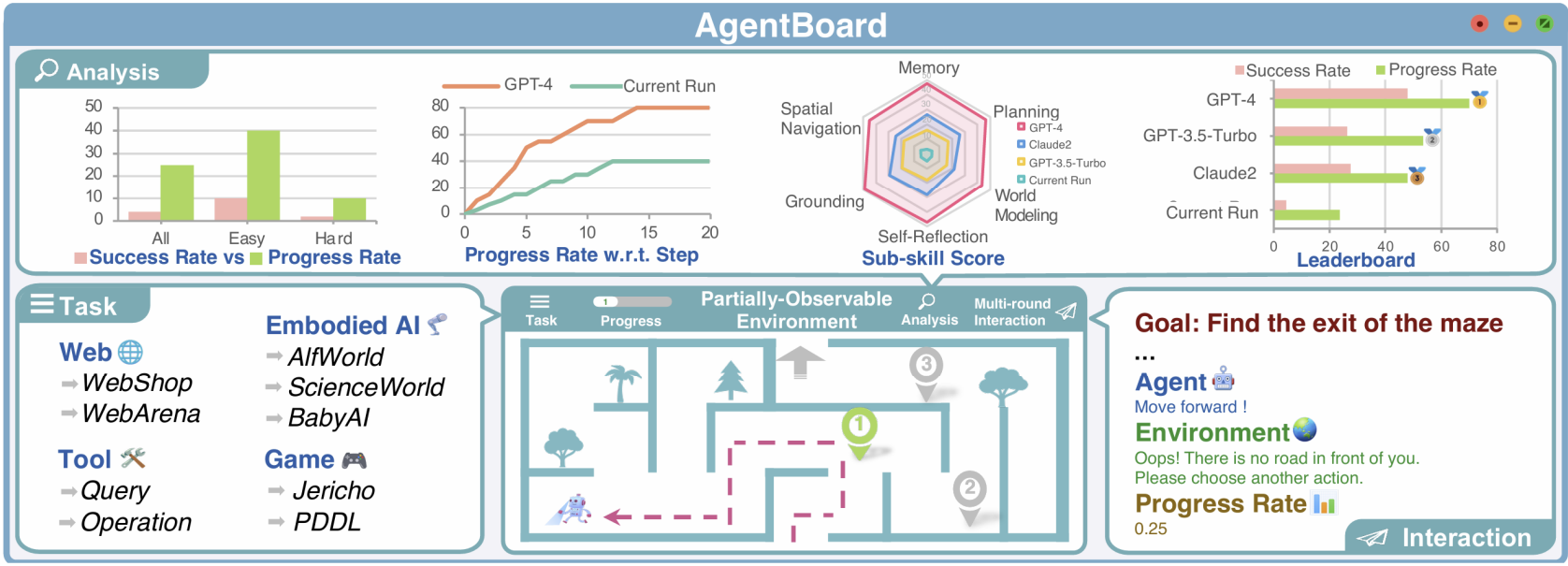


Figure 1: The illustrative overview of AgentBoard. It consists of a 9 diverse tasks. Agents interact in multi-rounds with partially-observable environments to achieve each subgoal. Furthermore, AgentBoard provides an open-source analytical evaluation framework, as shown in the figure.

[Ma et al "AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents", **NeurIPS 2024**]

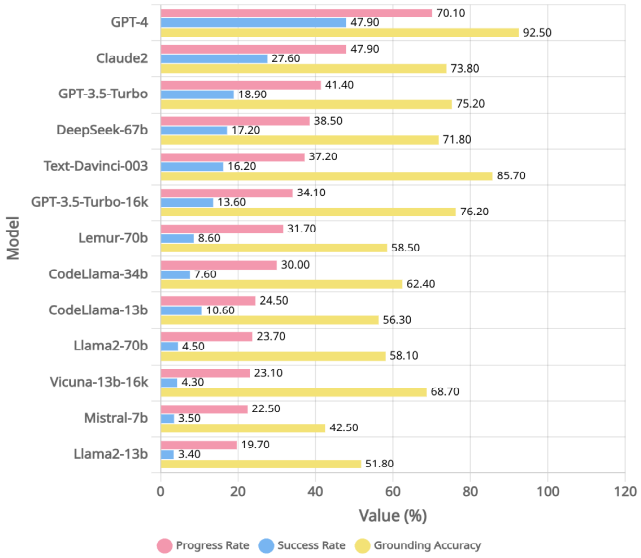
Agent Benchmarking

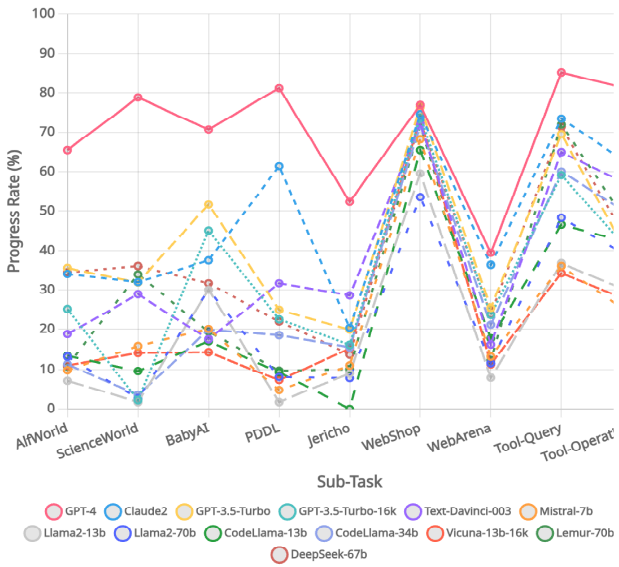
Juan Carlos Niebles, Salesforce

AgentBoard: evaluation framework

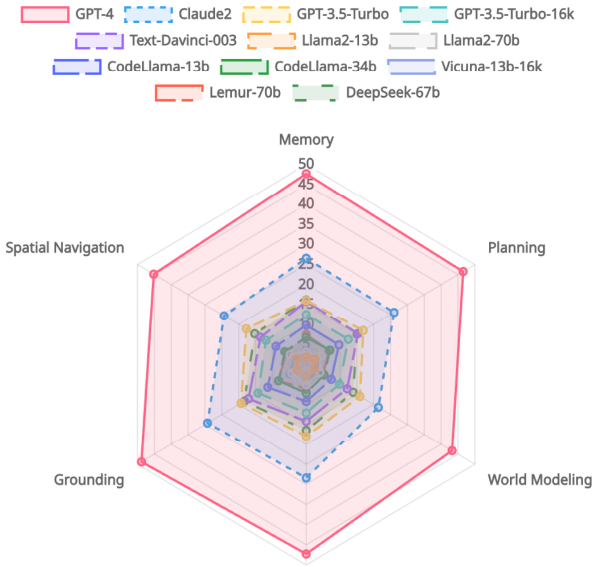
https://github.com/hkust-nlp/AgentBoard

Main Performance





Dimensional Analysis



[Ma et al "AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents", **NeurIPS 2024**]

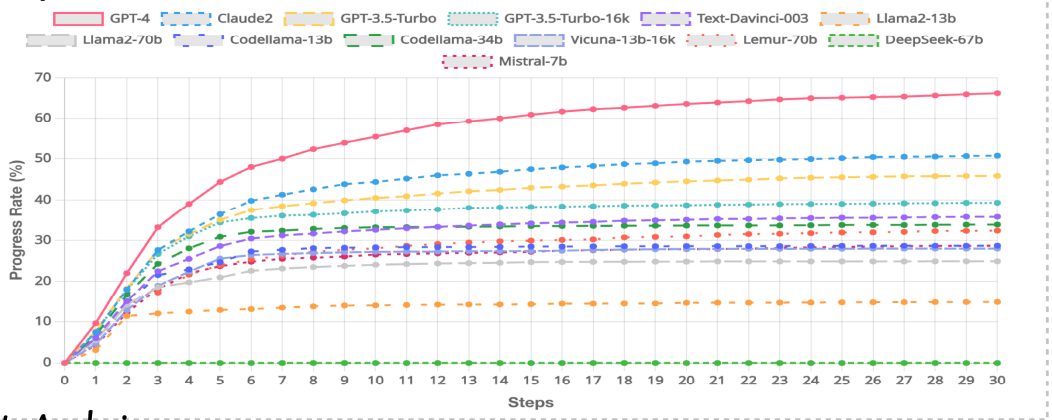
Agent Benchmarking

Juan Carlos Niebles, Salesforce

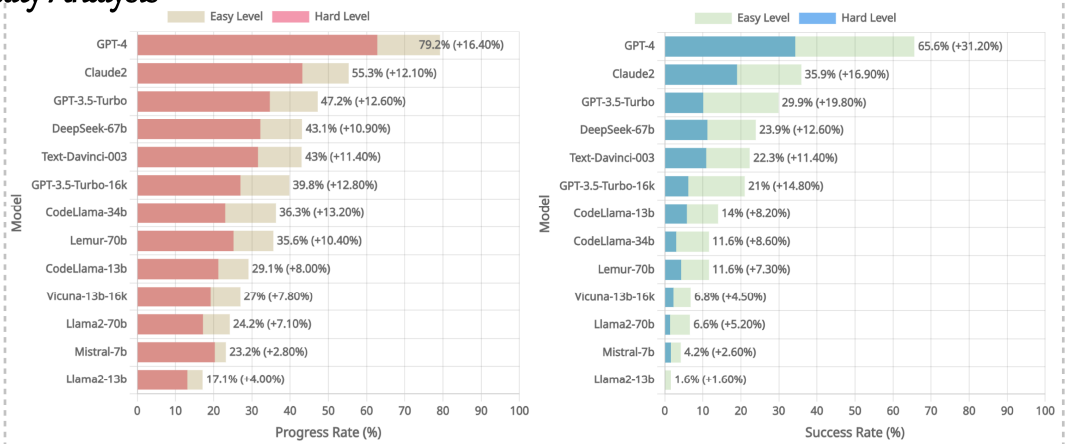
AgentBoard: evaluation framework

https://github.com/hkust-nlp/AgentBoard

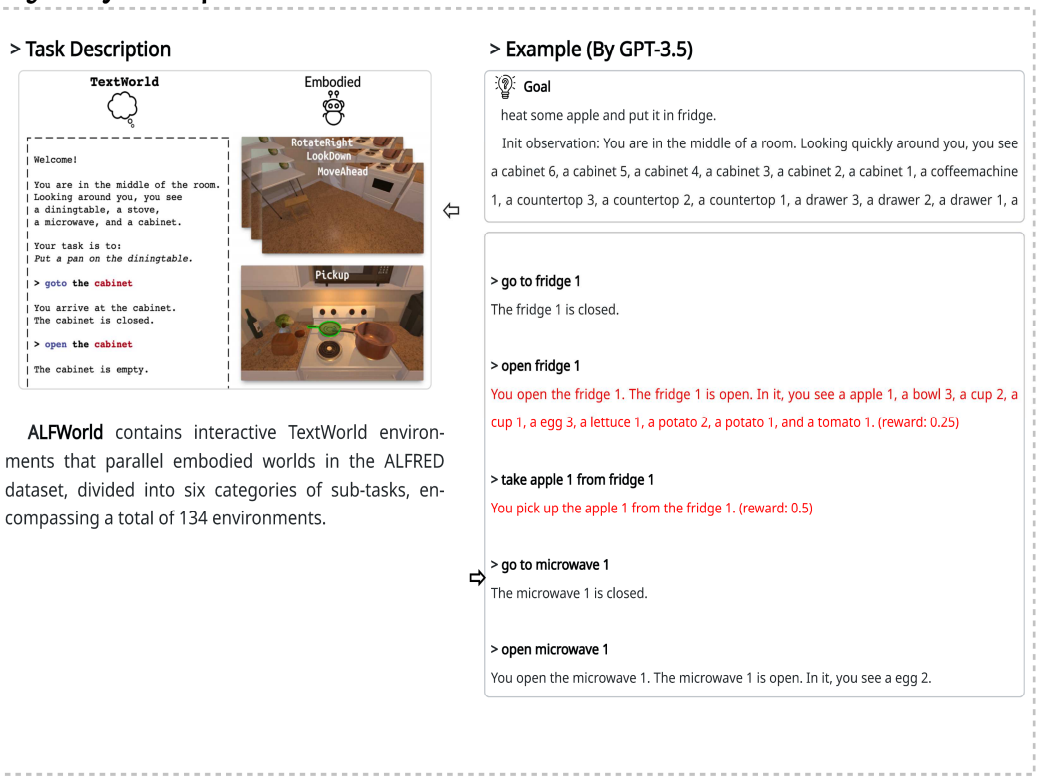
Steps Analysis



Difficulty Analysis



Trajectory Example



[Ma et al “AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents”, **NeurIPS 2024**]

Agent Architectures

How are we building Agents today?



Agent Architectures

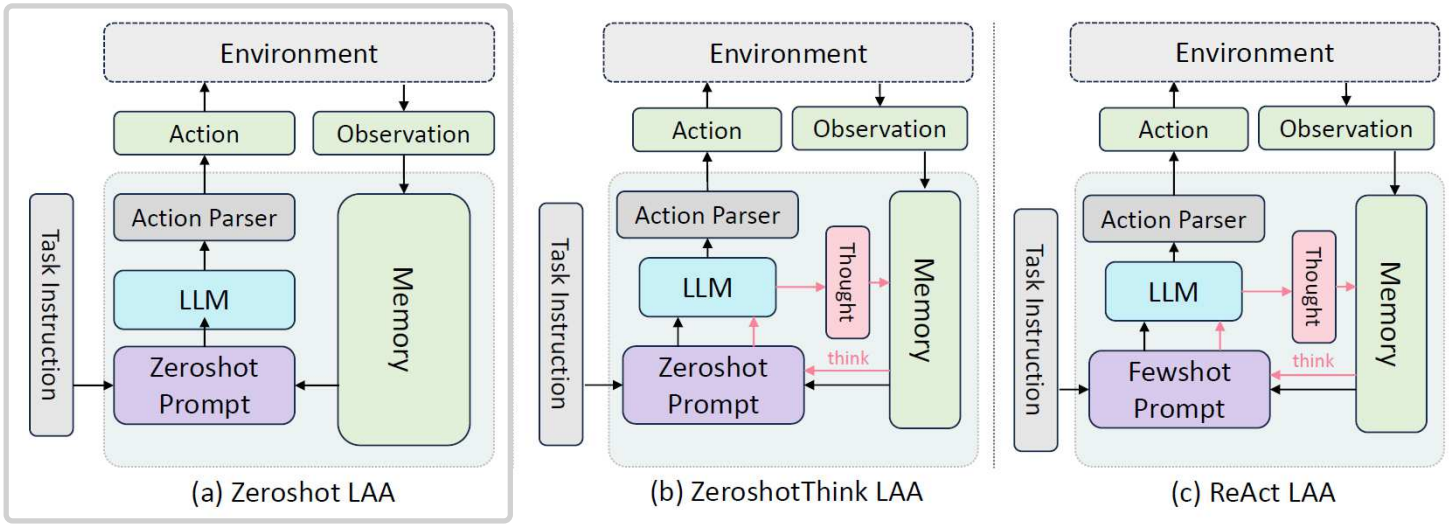
❖ LLM-augmented Autonomous Agents (LAAs)

LLM-augmented Autonomous Agents (LAAs) is able to **generate actions with its core LLM** and **interact with environments**, which facilitates the ability to resolve complex tasks by conditioning on past interactions such as *observations* and *actions*

[Zhiwei Liu. “BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents”. **ICLR 2024** Workshop on LLM Agents]

Agent Architectures

❖ LLM-augmented Autonomous Agents (LAAs) ; Solo Agents



Zeroshot LAA (ZS-LAA) directly extends the LLM to be action executor.

Ex: If we prompt LAA to understand the click action with "click: using this action to click observed [button], the clickable buttons are in [].", it may behave as a web navigation agent.

Zeroshot LAA [Initial step]

① Firstly, the ZS-LAA receives the **task instruction** and constructs the **zeroshot prompt**.

② **LLM layer** generates a possible **response**, which is parsed to output a feasible **action**.

③ The **observation** from **environment** is appended into the **agent memory**.

[Working steps]:

① The agent checks whether the task is finished.

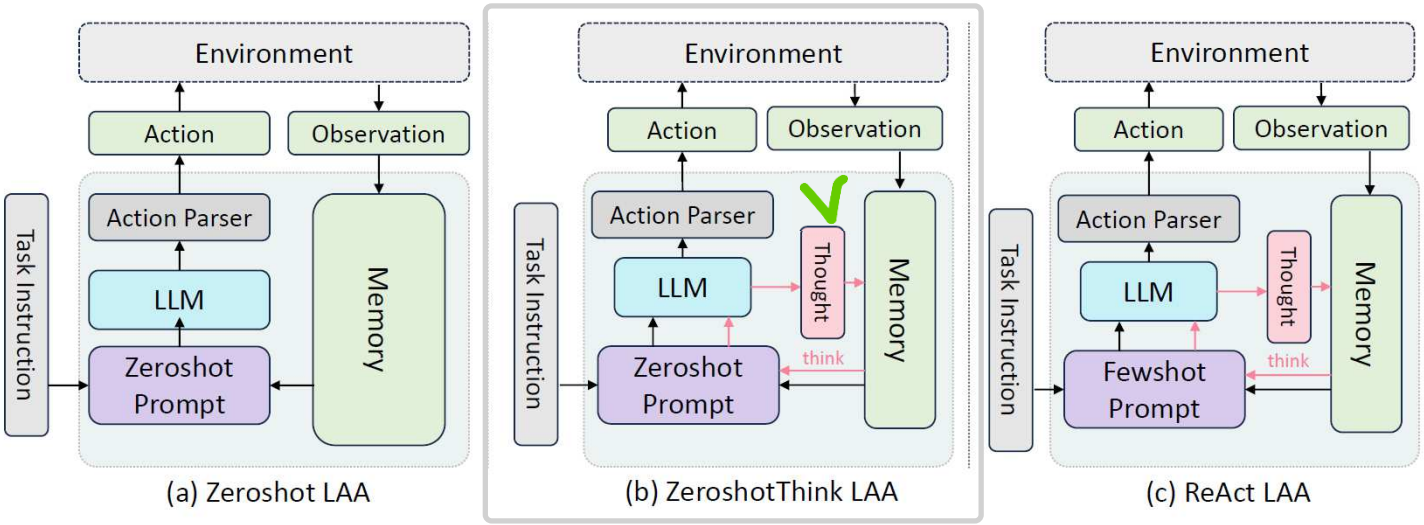
② If not, ZS-LAA retrieves the previous actions and observations from memory, and constructs the prompts for LLM to generate the next executable actions.

③ ZS-LAA continues the working stage until reaching the maximum steps or completing the task.

[Zhiwei Liu. "BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents". **ICLR 2024** Workshop on LLM Agents]

Agent Architectures

❖ LLM-augmented Autonomous Agents (LAAs) ; Solo Agents



ZeroshotThink LAA (ZST-LAA) : Additional self-think flow

Self-think is running in intermediate steps of action generations flow, which enables the Chain-of-Thought (CoT) reasoning ability

ZeroshotThink LAA [Self-think Step]

① Before generating the next action, ZST-LAA collect observations and previous actions to construct the think prompt.

② Then, the thought is stored into memory.

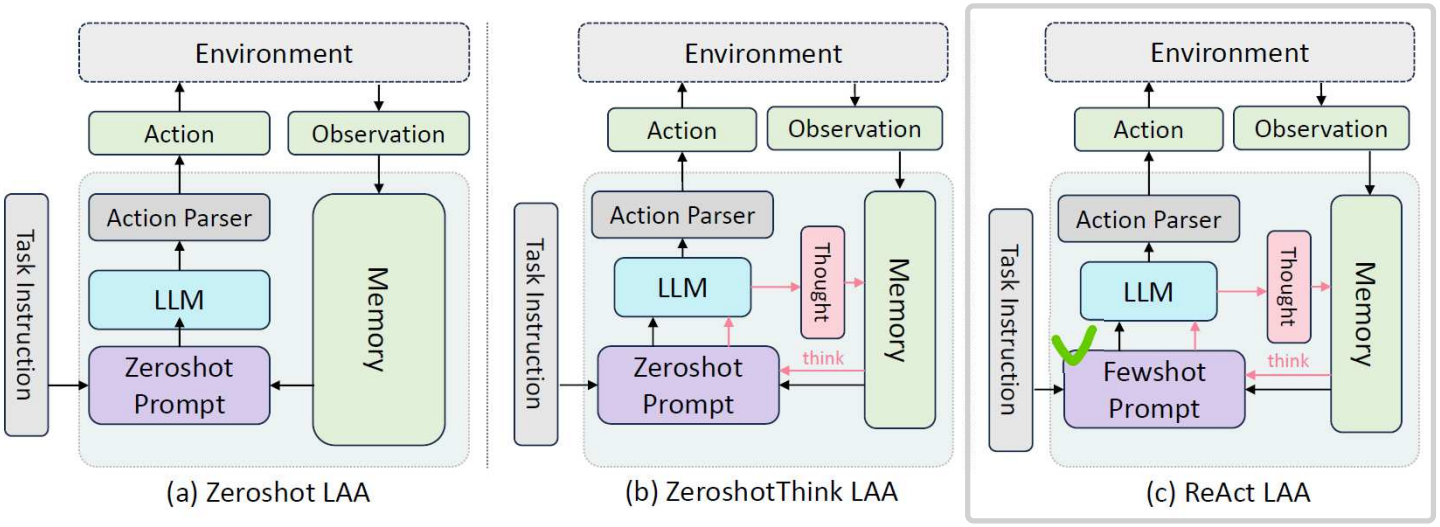
Self-think step is generally useful when given reasoning tasks.

Note that the **think prompt** is **also in a zero-shot format**, such as "**think: using this action to plan your actions and reasoning**".

[Zhiwei Liu. "BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents". **ICLR 2024** Workshop on LLM Agents]

Agent Architectures

❖ LLM-augmented Autonomous Agents (LAAs) ; Solo Agents



ReAct LAA : additionally advances ZST-LAA in the prompt layer, where fewshot examples are provided

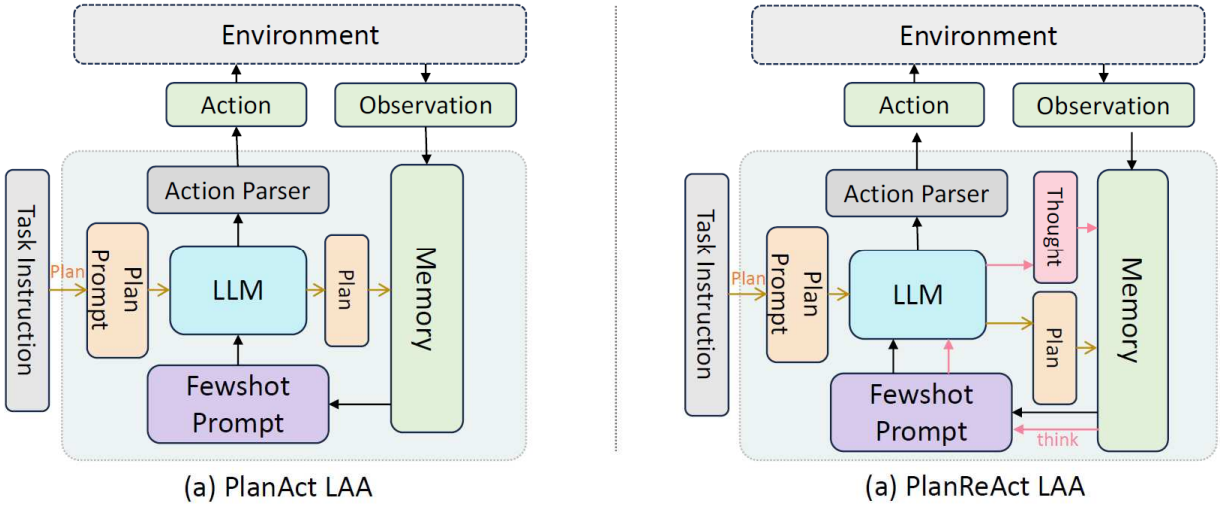
ReAct LAA is able to leverage **successful running examples** to **improve the action generation ability of LLM** and **enhance the environment interaction of LAA**, because those fewshot examples endows the in-context learning ability of LLM.

However, the drawback for ReAct LAA is that, due to the **limited context length**, **fewer token spaces are available after the occupancy of fewshot examples in the prompt**.

[Zhiwei Liu. “BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents”. **ICLR 2024** Workshop on LLM Agents]

Agent Architectures

❖ LLM-augmented Autonomous Agents (LAAs) ; Solo Agents



(a) PlanAct LAA

(a) PlanReAct LAA

PlanAct LAA is designed to facilitate the **planning ability of LAA**.

PlanAct LAA differs from ZSLAA in two parts: 1) the **planning flow** and 2) the **fewshot p**rompt.

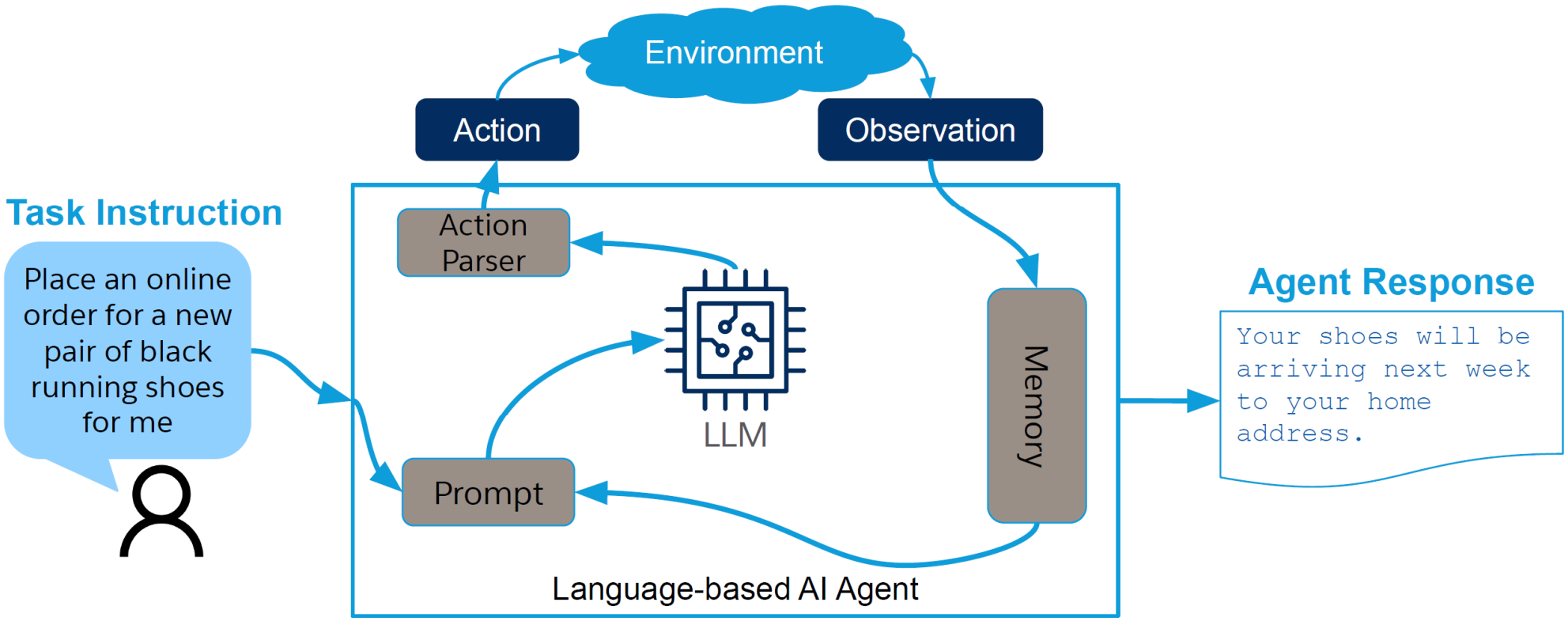
[Planning Step] PlanAct LAA generates a plan for a given task before interacting with environments. The plan is memorized and will be retrieved to construct prompts

PlanReAct LAA extends PlanAct LAA with additional **self-think flow**, which also enables the CoT ability. Since the Planning flow is executed before the LAA observes the environment, self-think flow alleviates the hallucination incurred from incorrect plans.

[Zhiwei Liu. “BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents”. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

High-level Language-based Agent framework



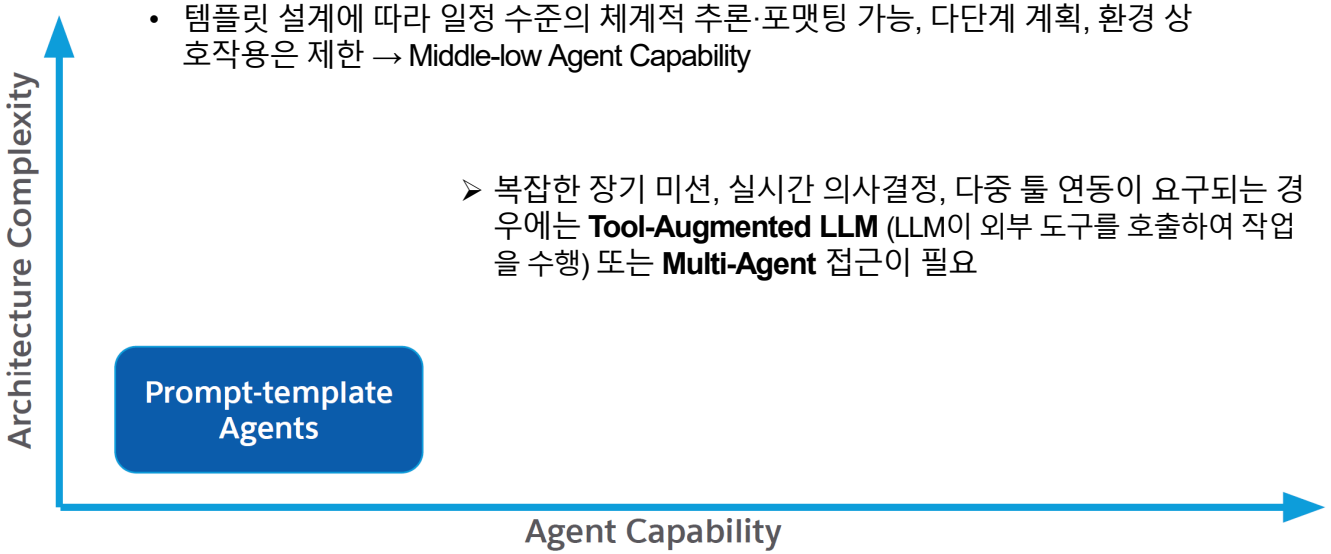
Juan Carlos Niebles, Salesforce

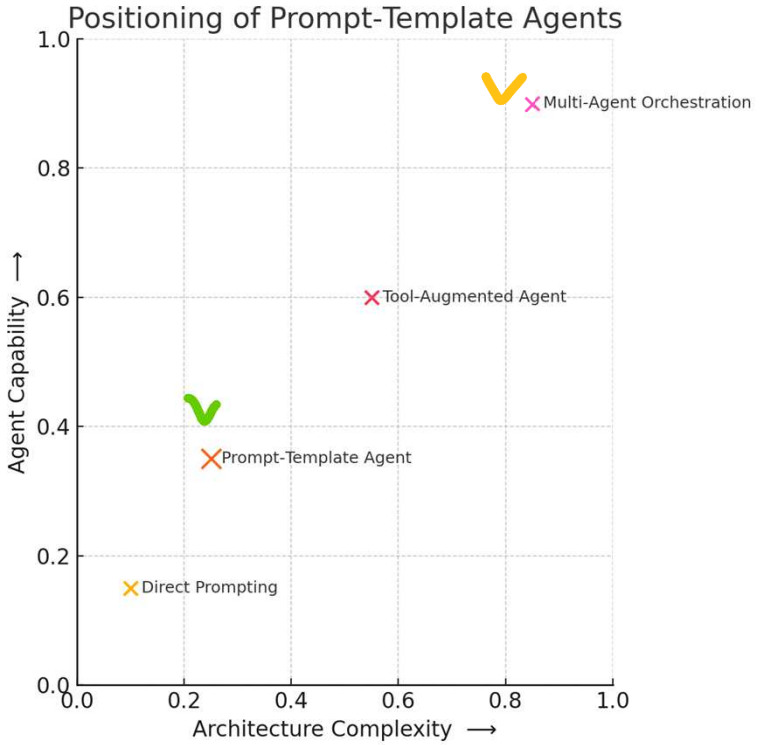
Language-based AI Agents

➢ **Prompt-Template Agent**

- 단일 LLM 호출, 미리 정의한 템플릿(**slot-filling**, system + few-shot prompt)만으로 컨텍스트 구성
- 외부 툴·메모리·에이전트 간 메시지 패싱 없음 → Low Architecture Complexity
- 템플릿 설계에 따라 일정 수준의 체계적 추론·포맷팅 가능, 다단계 계획, 환경 상호작용은 제한 → Middle-low Agent Capability

➢ 복잡한 장기 미션, 실시간 의사결정, 다중 툴 연동이 요구되는 경우에는 **Tool-Augmented LLM** (LLM이 외부 도구를 호출하여 작업을 수행) 또는 **Multi-Agent** 접근이 필요





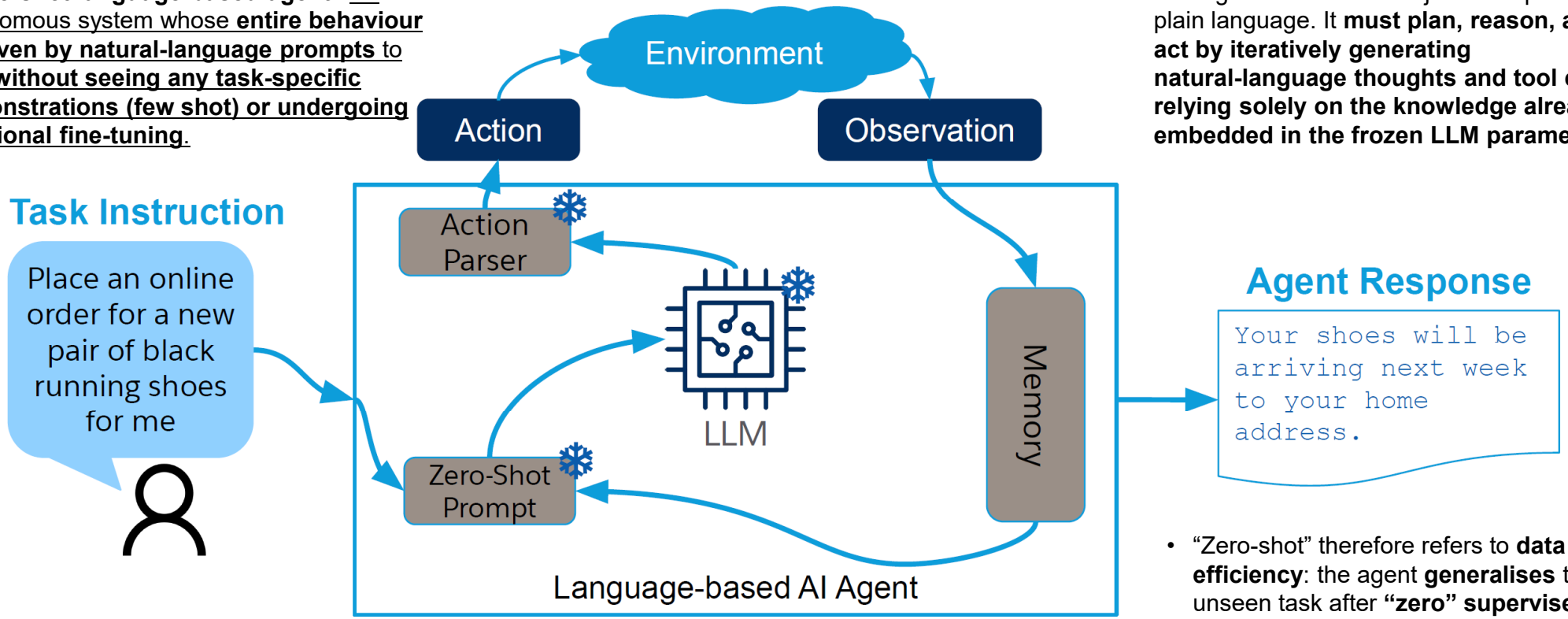
- Architecture Complexity : 에이전트 구현위한 필요 구성 요소 수, 모듈 간 상호작용, 외부 시스템 통합 정도
  - 낮음 : 단일 프롬프트·모델 호출 중심
  - 높음 : 멀티-에이전트, 툴 오케스트레이션, 지속 메모리
- Agent Capability : 문제 해결 범위·난이도, 자율성, 계획·추론·행동 능력
  - 낮음 : 단순 응답·변환 중심
  - 높음 : 동적 계획, 장기 목표 추적, 복합 도구 활용

[☞ ChatGPT o1]

Juan Carlos Niebles, Salesforce

Zero-shot Language-based Agent

A zero-shot language-based agent : An autonomous system whose **entire behaviour is driven by natural-language prompts** to LLM **without seeing any task-specific demonstrations (few shot) or undergoing additional fine-tuning**.



- The agent receives an objective expressed in plain language. It **must plan, reason, and act by iteratively generating natural-language thoughts and tool calls, relying solely on the knowledge already embedded in the frozen LLM parameters**.

- "Zero-shot" therefore refers to **data efficiency**: the agent **generalises** to an unseen task after **"zero" supervised examples**.

[Zhiwei Liu. "BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents". **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

Zero-shot Agent Example

System Prompt

You are an intelligent agent. You should follow your [Role], [Action_Doc] to take actions. Your generation should follow the example format. Finish the task as best as you can.

[Role] You can interact with the webshop. [End of Role]

[Constraint] Your generation should be simple and clear. [End of Constraint]

[Action_Doc] [{'name': **'search**', 'description': '<fc_description>', 'parameters': {**'product**': '<param_description>'}},
{'name': **'click**', 'description': '<fc_description>', 'parameters': {**'button**': '<param_description>'}},
{'name': **'Finish**', 'description': '<fc_description>', 'parameters': {**'response**': '<param_description>'}}] [End of Action_Doc]

Using the following action format example to generate well formatted actions.

[ActionFormatExample]
Action:**search**[{"**product**": "the name of the product to search for"}]
Action:**Finish**[{"**response**": "this is the finish action response. Respond towards the task instruction."}]
Action:**click**[{"**button**": "the name of the button to click"}]
[End of ActionFormatExample]

[Zhiwei Liu. “BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents”. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

Zero-shot Agent Example

Task Instruction: I need a long clip-in hair extension which is natural looking, and price lower than 40.00 dollars

[Execution]

Action: search[{"product": "long clip-in hair extension"}]

Action: click[{"button": "B09C337K8S"}]

Action:
Finish[{"response": "I found a long clip-in hair extension that is natural looking and priced lower than $40.00. It is the Hair Clip Extensions for Women Natural Looking Hair Extensions Clip in Hairpieces Black Synthetic Hair Extensions, priced at $29.99."}]

[Environment]

Observation: WEB PAGE: {
[Back to Search] Page 1 (Total results: 15)
[Next >]
[B09C337K8S]
Hair Clip Extensions for Women Natural Looking Hair Extensions clip in Hairpieces Black Synthetic Hair Extensions $29.99
[B099K9Z9L2]
Clip in Hair Extensions 20" Double Weft 100% Remy Human Hair Grade 8A Quality 7pcs 16clips 70grams $43.99
}

Observation: WEB PAGE: {
[Back to Search]
[< Prev]
Hair Clip Extensions for Women Natural Looking Hair Extensions Clip in Hairpieces Black Synthetic Hair Extensions Price: $29.99
Rating: N.A.
[Buy Now]
}

[Zhiwei Liu. "BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents". **ICLR 2024** Workshop on LLM Agents]

Agent Architectures

Juan Carlos Niebles, Salesforce

Zero-shot Agent WebShop Results: Average Reward

- Bigger LLMs tend to achieve better agent performance out of the box
- Some LLMs struggle in Agent tasks in the zero-shot setting
- Big gap open source <-> GPT models

Table 1: Average reward in the WebShop environment. Len denotes the maximum context length. **Bold** results denote the best results in one row, *i.e.* best LAA architecture w.r.t. one LLM. Underline results denote the best performance in one column, *i.e.* best LLM regarding one LAA architecture.

LLM	Len.	LAA Architecture					
		ZS	ZST	ReAct	PlanAct	PlanReAct	BOLAA
fastchat-t5-3b	2k	0.3971	0.2832	0.3098	0.3837	0.1507	**0.5169**
vicuna-7b	2k	0.0012	0.0002	**0.1033**	0.0555	0.0674	0.0604
vicuna-13b	2k	0.0340	0.0451	0.1509	0.3120	0.4127	**0.5350**
vicuna-33b	2k	0.1356	0.2049	0.1887	0.3692	0.3125	**0.5612**
llama-2-7b	4k	0.0042	0.0068	0.1248	0.3156	0.2761	**0.4648**
llama-2-13b	4k	0.0662	0.0420	0.2568	**0.4892**	0.4091	0.3716
llama-2-70b	4k	0.0122	0.0080	0.4426	0.2979	0.3770	**0.5040**
mpt-7b-instruct	8k	0.0001	0.0001	0.0573	0.0656	**0.1574**	0.0632
mpt-30b-instruct	8k	0.1664	0.1255	0.3119	0.3060	0.3198	**0.4381**
xgen-8k-7b-instruct	8k	0.0001	0.0015	0.0685	0.1574	0.1004	**0.3697**
longchat-7b-16k	16k	0.0165	0.0171	0.069	0.0917	0.1322	**0.1964**
longchat-13b-16k	16k	0.0007	0.0007	0.2373	0.3978	**0.4019**	0.3205
text-davinci-003	4k	0.5292	0.5395	0.5474	0.4751	0.4912	**0.6341**
gpt-3.5-turbo	4k	0.5061	0.5057	0.5383	0.4667	0.5483	**0.6567**
gpt-3.5-turbo-16k	16k	0.5657	0.5642	0.4898	0.4565	0.5607	**0.6541**

Zero-shot Agent HotPotQA Results

- GPT models ahead again

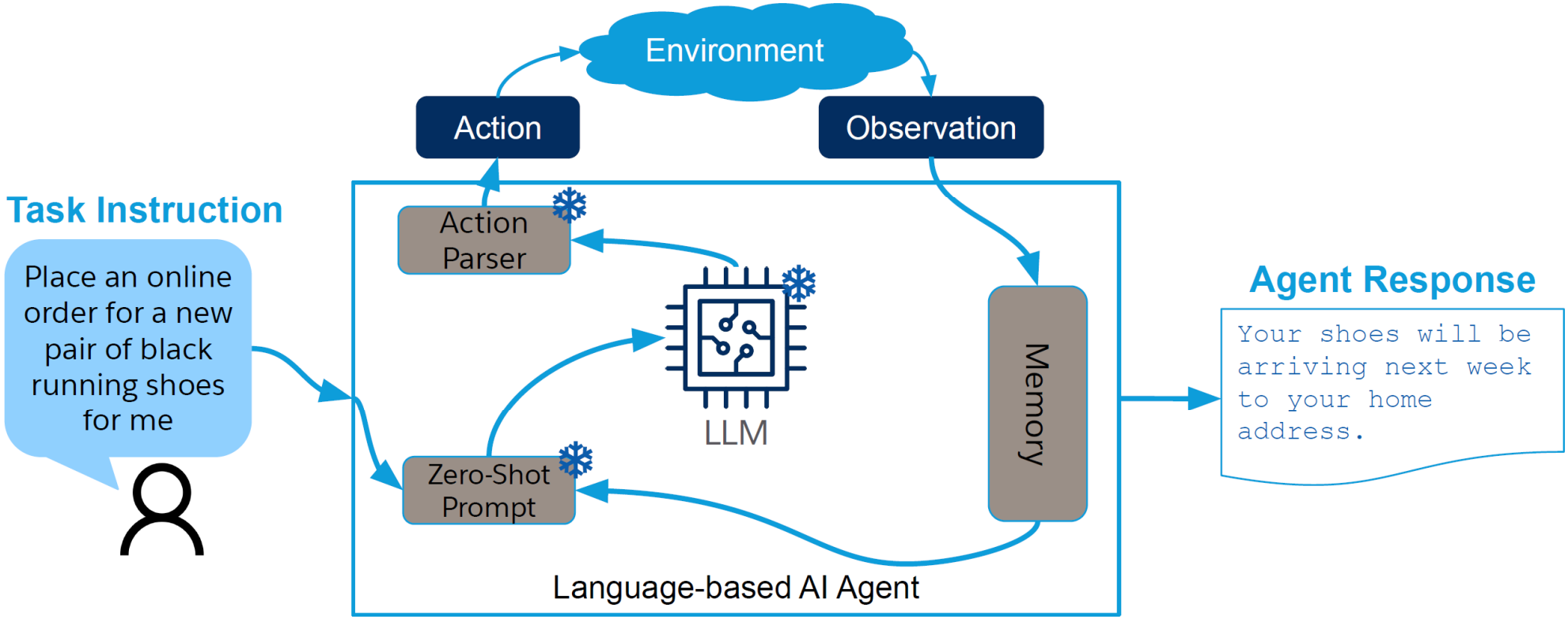
Table 2: Average recall in the WebShop environment. Len denotes the maximum context length. **Bold** results denote the best results in one row, *i.e.* best LAA architecture w.r.t. one LLM. Underline results denote the best performance in one column, *i.e.* best LLM regarding one LAA architecture.

LLM	Len.	LAA Architecture					
		ZS	ZST	ReAct	PlanAct	PlanReAct	BOLAA
fastchat-t5-3b	2k	0.3533	0.3122	0.3800	0.3700	0.3722	**0.3867**
vicuna-7b	2k	0.0833	0.0500	0.3600	0.3233	0.3278	**0.3522**
vicuna-13b	2k	0.0867	0.0644	0.3622	0.3444	0.2367	**0.3700**
vicuna-33b	2k	0.3600	0.3411	0.3822	0.3733	0.3567	**0.3956**
llama-2-7b	4k	0.0678	0.0311	0.3744	0.3400	0.3578	**0.3856**
llama-2-13b	4k	0.2856	0.2211	0.3844	0.3278	0.3500	**0.4078**
llama-2-70b	4k	0.3344	0.3244	0.3789	0.3400	0.3600	**0.4011**
mpt-7b-instruct	8k	0.0144	0.0322	**0.3644**	0.3200	0.3400	0.3600
mpt-30b-instruct	8k	0.2973	0.3372	0.3333	0.3575	0.3412	**0.3900**
xgen-8k-7b-instruct	8k	0.0667	0.1400	0.3711	0.3400	0.3278	**0.3800**
longchat-7b-16k	16k	0.1344	0.1856	0.3644	0.3622	0.3622	**0.3811**
longchat-13b-16k	16k	0.0756	0.0867	0.3678	0.3467	0.3471	**0.3789**
text-davinci-003	4k	0.3800	0.3856	0.3767	0.3711	0.3889	**0.3956**
gpt-3.5-turbo	4k	0.3889	0.3756	**0.3933**	0.3789	0.3867	0.3929
gpt-3.5-turbo-16k-0613	16k	0.3856	0.3833	**0.4011**	0.3756	0.3811	0.3933

[Zhiwei Liu. “BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents”. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

Zero-shot Language-based Agent

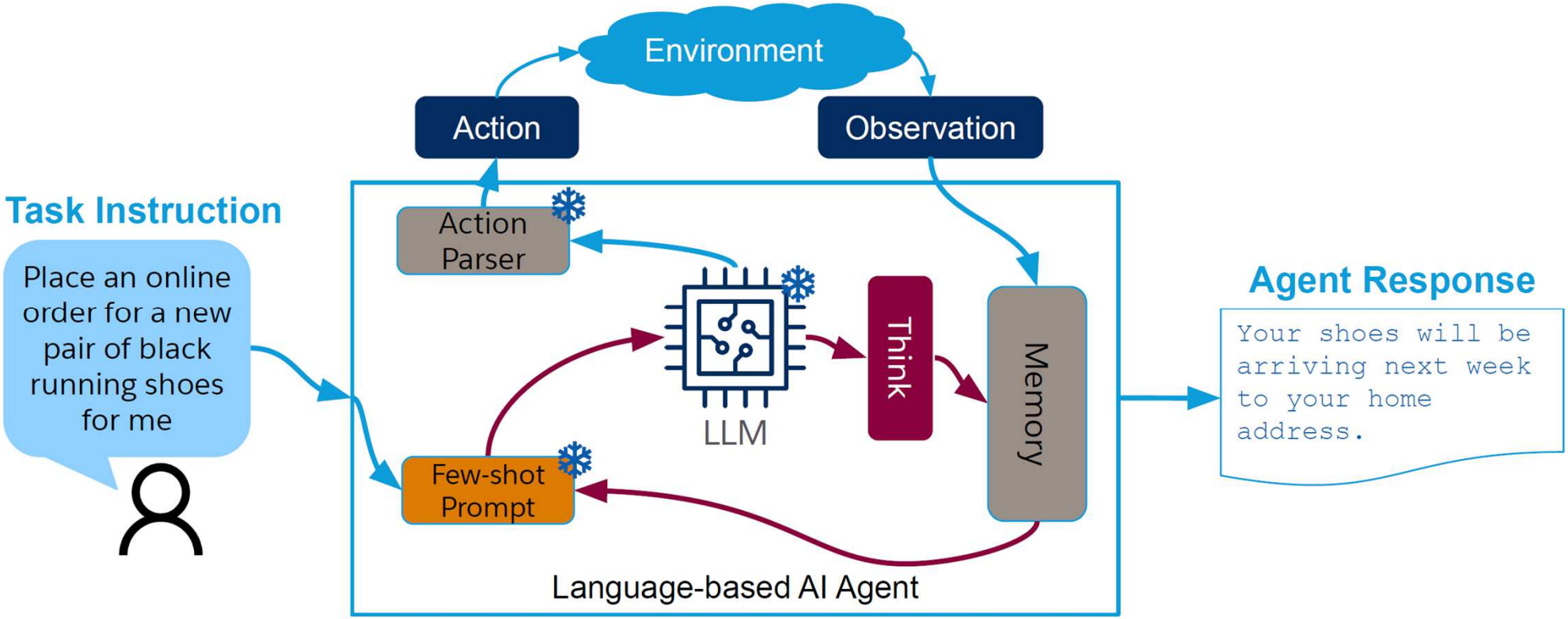


[Zhiwei Liu. “BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents”. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

ReAct Agent

https://github.com/ysymyth/ReAct



[Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. **ICLR 2023**]

Agent Architectures

ReAct

https://github.com/ysymyth/ReAct

- ReAct : Chain-of-Thought (CoT) 스타일의 자연어 추론(**Reasoning**)과 도구 호출(**Action**)을 프롬프트 수준에서 결합하는 프레임워크

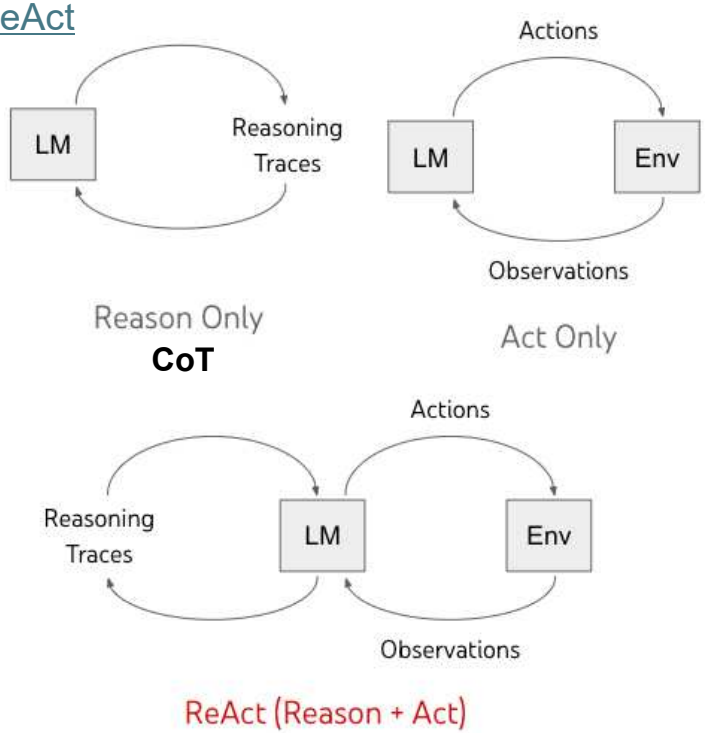
- **Reasoning**: (*내부추론*) 모델이 자신의 계획, 추론, 판단을 자연어로 기술
- **Acting**: (*도구호출*) 모델이 도구(tool)나 환경과 상호작용할 수 있는 명령을 생성
- **Observation**: (*외부 Feedback*) 도구에서 반환된 결과를 다시 LLM의 입력에 반영하여 루프를 지속

- ReAct 구조

  [질문]
  → Thought: 중간 추론
  → Action: 도구 호출 (예: 검색, 계산, DB 조회)
  → Observation: 도구 응답
  → Thought: 다음 추론
  → Action: ...
  → 최종 Answer 출력

- 이 과정은 여러 번 반복될 수 있으며, 모든 단계는 자연어 기반 텍스트로 처리됨
- 주로 **few-shot prompting**이나 **zero-shot prompting**으로 사용.

[Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. **ICLR 2023**]



Previous methods prompt language models (LM) to either generate self-conditioned reasoning traces or task-specific actions.
We propose ReAct, a new paradigm that combines reasoning and acting advances in language models.
[☞ https://research.google/blog/react-synergizing-reasoning-and-acting-in-language-models/]

Agent Architectures

ReAct

https://github.com/ysymyth/ReAct

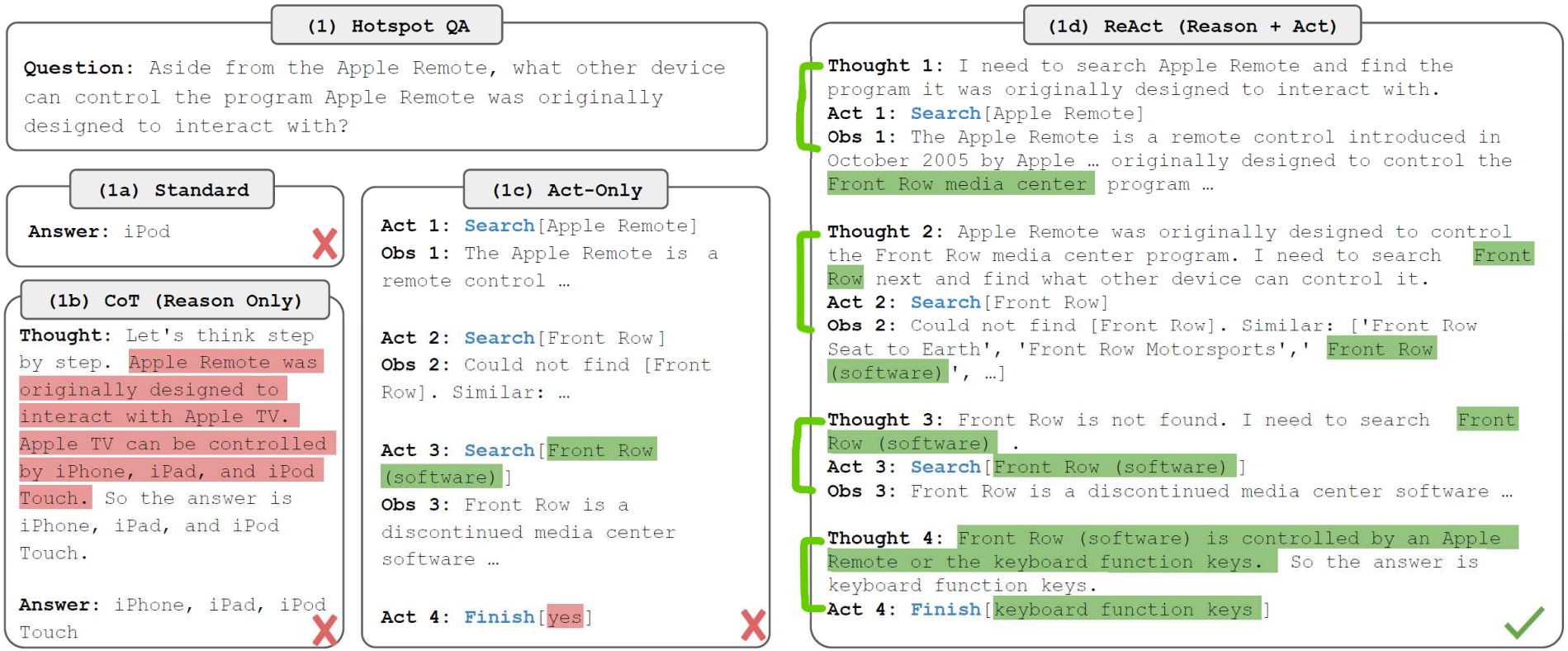


Figure 1: (1) Comparison of 4 prompting methods, (a) Standard, (b) Chain-of-thought (CoT, Reason Only), (c) Act-only, and (d) ReAct (Reason+Act), solving a HotpotQA (Yang et al., 2018) question

[Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. **ICLR 2023**]

Agent Architectures

ReAct

https://github.com/ysymyth/ReAct

(2) AlfWorld

```
You are in the middle of a room. Looking quickly around
you, you see a cabinet 6, a cabinet 1, a coffee machine 1,
a countertop 3,  a stove burner 1, and a toaster 1.
Your task is to: Put some pepper shaker on a drawer.
```

(2a) Act-Only

```
Act 1: Go to drawer 1
Obs 1: The drawer 1 is closed.

Act 2: Open drawer 1
Obs 2: You open the drawer 1. The drawer 1 is open. In it,
you see a dishsponge 2, and a spoon 1.

Act 3: Go to sinkbasin 1
Obs 3: On the sinkbasin 1, you see a dishsponge 3, a
spatula 1, and a spoon 2.

Act 4: Take peppershaker 1 from sinkbasin 1
Obs 4: Nothing happens.

Act 5: Take peppershaker 1 from sinkbasin 1
Obs 5: Nothing happens.
```

✗

(2b) ReAct (Reason + Act)

```
Act 1: Think[First I need to find a pepper shaker...more
likely to appear in cabinets (1-6), countertops (1-3), ...   ]
Act 2: Go to cabinet 1
Obs 2: On the cabinet 1, you see a vase 2.
(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)

Act 6: Go to countertop 3
Obs 6: On the countertop 3, you see a apple 1, a bread 1,
..... a pepper shaker 1, and a vase 3.

Act 7: Take pepper shaker 1 from countertop 3
Obs 7: You pick up the pepper shaker 1 from the countertop 3.

Act 8: Think[Now I find a pepper shaker 1. Next, I need to
put it in/on drawer 1. ]
Act 9: Go to drawer 1
Obs 9: Drawer 1 is closed.

Act 10: Open drawer 1
Obs 10: You open Drawer 1 …

Act 11: Put pepper shaker 1 in/on drawer 1
Obs 11: You put pepper shaker 1 in/on the drawer 1.
```

✓

Figure 1: (2) Comparison of (a) Act-only and (b) ReAct prompting to solve an AlfWorld (Shridhar et al., 2020b) game. In both domains, we omit in-context examples in the prompt, and only show task solving trajectories generated by the model (Act, Thought) and the environment (Obs).

[Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. **ICLR 2023**]

Juan Carlos Niebles, Salesforce

ReAct Agent Example

System Prompt

You are an intelligent agent. You should follow your [**Role**], [**Action_Doc**] to take actions. Your generation should follow the example format. Finish the task as best as you can.

[**Role**] You can interact with the webshop. [**End of Role**]

[**Constraint**] You generation should be simple and clear. [**End of Constraint**]

[**Action_Doc**] [{'name': **'search**', 'description': 'search for a product in the webshop', 'parameters': {**'product**': 'the name of the product to search for'}},
{'name': **'Finish**', 'description': 'Complete the task with a response.', 'parameters': {**'response**': 'this is the finish action response. Respond towards the task instruction.'}},
{'name': **'click**', 'description': 'click a button in the web page', 'parameters': {**'button**': 'the name of the button to click'}},
{'name': **'Think**', 'description': **'Conduct thinking and reasoning process for solving task**.', 'parameters': {'**response**': '**this is your thinking response. Be specific and critical**.'}}]
[**End of Action_Doc**]

Using the following action format example to generate well formatted actions.

[ActionFormatExample]
Action:**search**[{"**product**": "the name of the product to search for"}]
Action:**Finish**[{"**response**": "this is the finish action response. Respond towards the task instruction."}]
Action:**click**[{"**button**": "the name of the button to click"}]
[End of ActionFormatExample]

[Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. **ICLR 2023**]

Juan Carlos Niebles, Salesforce

ReAct Agent Example

[Example]

Task: i would like a 3 ounce bottle of bright citrus deodorant for sensitive skin, and price lower than 50.00 dollars

Action: Think[{"**response**": "I should first use **Search** to search bright citrus deodorant for sensitive skin."}]

Observation: OK

Action: **search**[{"**product**": "bright citrus deodorant for sensitive skin"}]

Observation: WEB PAGE: {

[Back to Search]

[< Prev]

scent [assorted scents][bright citrus][calming lavender] size [3 ounce (pack of 1)]}

Action: **click**[{"**button**": "bright citrus"}]

Observation: "You have clicked bright citrus.

Action: **click**[{"**button**": "3 ounce (pack of 1)"}]

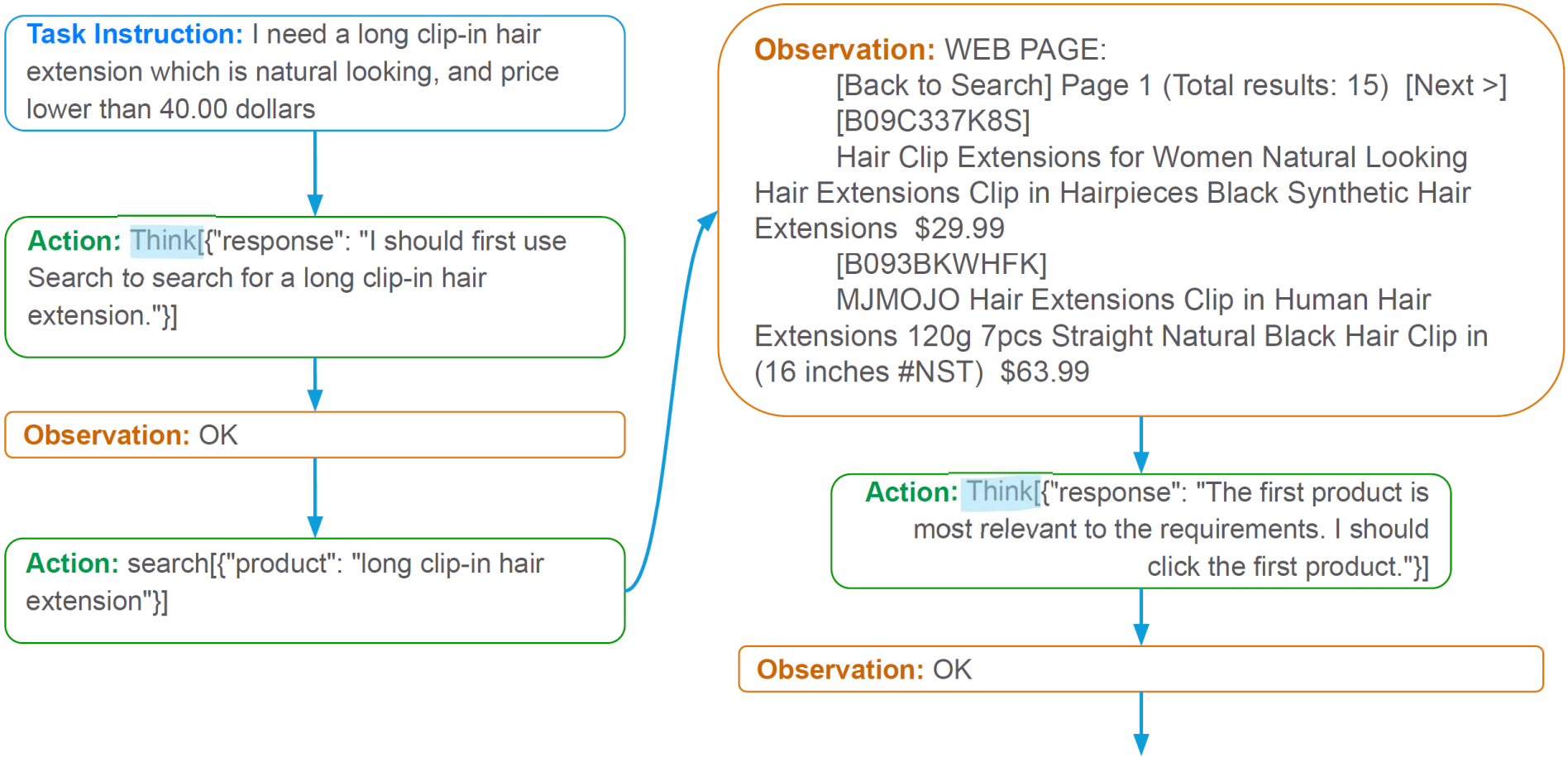
Observation: "You have clicked 3 ounce (pack of 1).

Action: **click**[{"**button**": "Buy Now"}]

[Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. **ICLR 2023**]

Juan Carlos Niebles, Salesforce

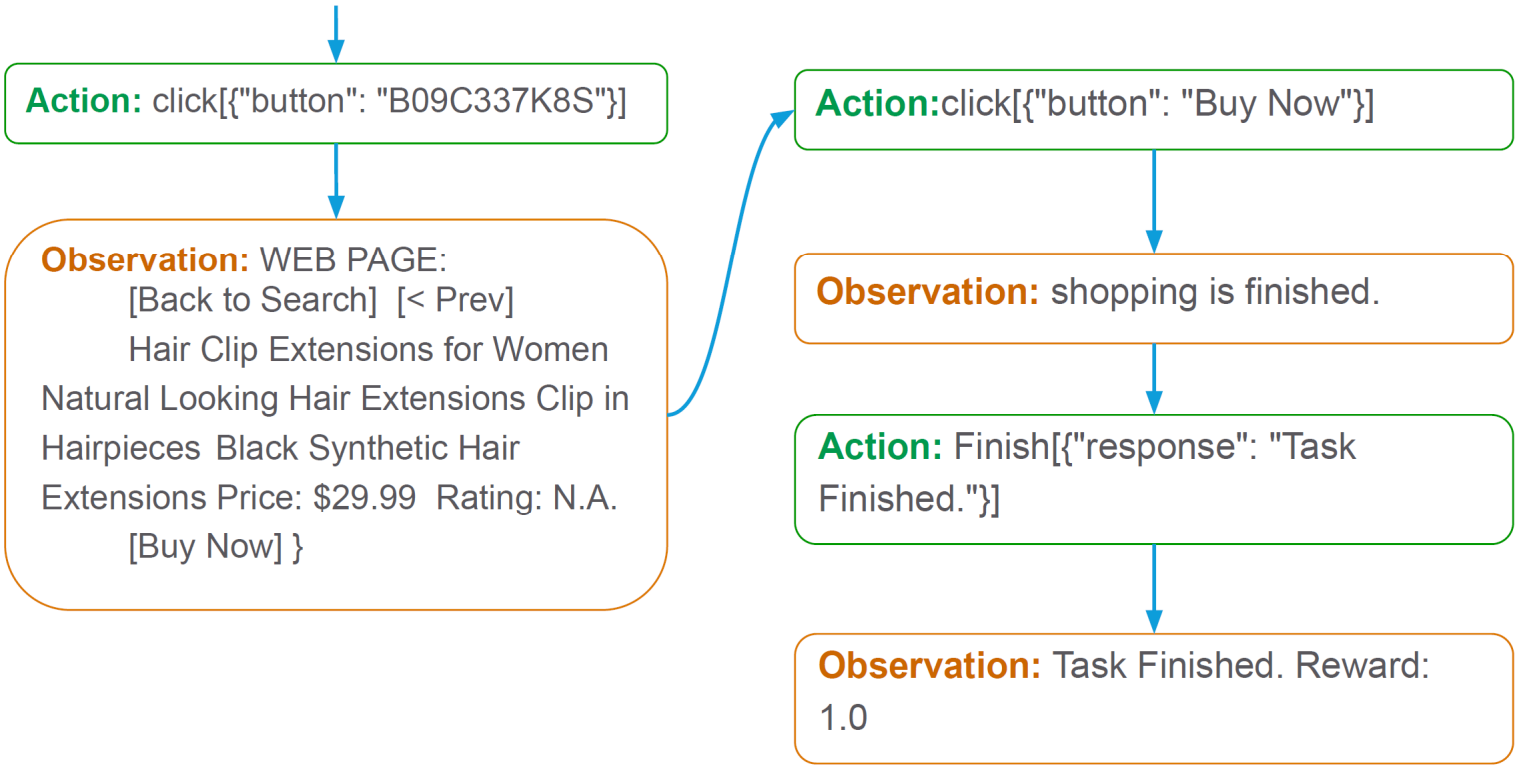
ReAct Agent Example



[Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. **ICLR 2023**]

Juan Carlos Niebles, Salesforce

ReAct Agent Example



[Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. **ICLR 2023**]

Juan Carlos Niebles, Salesforce

ReAct Agent WebShop Results

- ReAct helps smaller models boost their performance.
- Big models/GPTs do not benefit much

LLM	Length	Zero-Shot	ReAct
fastchat-t5-3b	2k	0.40	**0.31**
vicuna-7b	2k	0.00	0.10
vicuna-13b	2k	0.03	0.15
vicuna-33b	2k	0.14	0.19
llama-2-7b	4k	0.00	0.12
llama-2-13b	4k	0.07	0.26
llama-2-70b	4k	0.01	**0.44**
mpt-7b-instruct	8k	0.00	0.06
mpt-30b-instruct	8k	0.17	**0.31**
xgen-8k-7b-instruct	8k	0.00	0.07
longchat-7b-16k	16k	0.02	0.07
longchat-13b-16k	16k	0.00	0.24
text-davinci-003	4k	**0.53**	**0.55**
gpt-3.5-turbo	4k	0.51	**0.54**
gpt-3.5-turbo-16k	16k	**0.57**	0.49

ReAct Agent HotPotQA Results

- ReAct helps smaller models boost their performance.
- Big models/GPTs do not benefit much.
- Model size more important than context length

LLM	Length	Zero-Shot	ReAct
fastchat-t5-3b	2k	0.03	0.07
vicuna-7b	2k	0.13	0.03
vicuna-13b	2k	0.15	0.26
vicuna-33b	2k	0.22	0.26
llama-2-7b	4k	0.04	0.26
llama-2-13b	4k	0.17	0.25
llama-2-70b	4k	**0.28**	**0.36**
mpt-7b-instruct	8k	0.10	0.17
mpt-30b-instruct	8k	0.16	0.33
xgen-8k-7b-instruct	8k	0.15	0.19
longchat-7b-16k	16k	0.08	0.22
longchat-13b-16k	16k	0.11	0.24
text-davinci-003	4k	**0.34**	**0.45**
gpt-3.5-turbo	4k	0.33	0.32
gpt-3.5-turbo-16k-0613	16k	0.30	0.19

[Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. **ICLR 2023**]

Juan Carlos Niebles, Salesforce

ReAct Agent

- **ReAct Features**

Items	Description
🧠 **추론+행동의 통합**	CoT만 사용하는 방식보다 더 **능동적이며 상호작용적인 문제 해결** 가능
🔁 **동적 재계획 가능**	Observation에 따라 Thought와 Action을 **실시간으로 조정**
🔍 **도구 사용 확장성**	검색엔진, 계산기, 코드 실행기, API 등 다양한 외부 도구와 결합 용이
💬 **추론 과정의 가시성**	step-by-step 로그로 모델의 **결정 과정을 해석**하기 쉬움
🧪 **탐색 기반 문제에 강함**	예: Quest 기반 질의, 체스 등에서 다단계 사고가 가능

- **ReAct Limitations**

Items	Description
🧩 **복잡도 증가**	프롬프트 길이가 늘어나고, Action/Observation의 흐름이 복잡한 오케스트레이션 필요
🧠 **Reasoning 오류 → 잘못된 Action**	초기에 잘못된 추론이 있으면 엉뚱한 도구 호출로 이어질 수 있음
⏳ **시간/비용 증가**	외부 도구 호출이 많아지면 응답 지연 및 비용 상승
🧱 **프롬프트 관리 어려움**	Observation이 누적되면서 프롬프트가 너무 커져 토큰 한계에 도달할 수 있음
🧪 **도구 정확도에 의존**	잘못된 Observation에 기반해 또 다른 오류를 양산할 수 있음

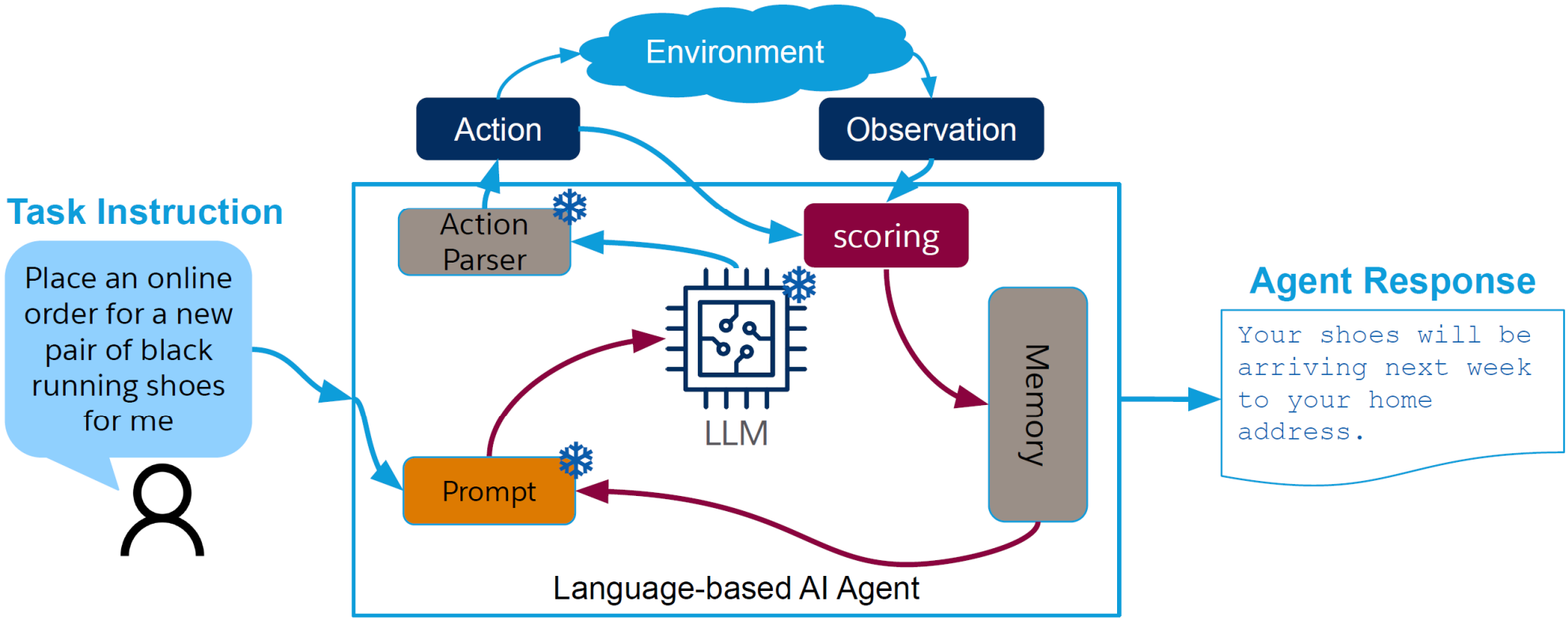
[☞ ChatGPT o1]

Quest : 쿼리 인식형 KV 캐시 선택 알고리즘
Quest: Query-Aware Sparsity for Efficient Long-Context LLM Inference

[Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. **ICLR 2023**]

Juan Carlos Niebles, Salesforce

REX Agent Architecture



[Murthy, et al. “REX: Rapid Exploration and eXploitation for AI agents”. **ICLR 2024** LLM Agents Workshop.]

Juan Carlos Niebles, Salesforce

REX

- Rapid **Exploration** and **eXploitation** of **action space** : LLM 기반 에이전트의 **탐색(exploration)**과 **활용(exploitation) 균형**을 개선하기 위한 새로운 접근법

- **REX 핵심구성 요소**

- **Reward integration** : REX는 Agent의 의사결정 과정에 Reward을 추가하여, Agent가 다양한 Action을 시도하고 그 결과를 학습함.
- **Upper Confidence Bound (UCB)** : REX는 UCB Score를 통합하여, Agent가 Exploration과 Exploitation 사이의 균형을 효과적으로 유지할 수 있도록 함
- **Offline Behavior Utilization** : REX는 Logs에서의 Offline Behavior을 활용하여, 기존의 Foundation model과의 원활한 통합을 가능하게 하며, 모델의 fine-tuning 없이도 작동함.
- **UCL (UCB applied for Logits of LLM)** ; UCL 통해 LLM Logits에 체계적으로 영향을 미쳐 Actions 생성에 대한 향상된 제어 제공

❖ **UCB Score**; **Encourages the language model to explore different actions while favoring those with higher potential rewards**

$$\mathrm{UCB}(s,\ a) = \hat{Q}(s,\ a) + C * \sqrt{\frac{\ln N(s)}{N(s,\ a)}}$$

- s : state, a : the action taken at s .
- $N(s)$: the number of times agent has produced s in it's solution
- $N(s,a)$: the number of times the agent took an action a from a state s.
- C : Constant to balance the exploration vs. exploitation trade off.
- $\hat{Q}(s,a)$: the cumulative reward for taking an action a at state s

❖ **UCL Score**; **Update the loglikelihoods of tokens corresponding to actions for the current state in the language model**

$$\mathrm{UCL}(s,\ a) = B * \ln \frac{\mathrm{UCB}(s,\ a)}{K}$$

- By manipulating the token logits with UCL scores during generation, the agent is compelled to execute the action yielding the highest reward
- Constants B & K : Control the extent to which logits of the LLM are offset.

[Murthy, et al. "REX: Rapid Exploration and eXploitation for AI agents". **ICLR 2024** LLM Agents Workshop.]

Juan Carlos Niebles, Salesforce

REX

- Monte Carlo Tree Search (**MCTS**) as backbone of REX

1) **Selection** ; Traverses down the tree by selecting *actions* that balance exploration and exploitation
   - Uses a selection policy, such as the Upper Confidence Bound, to determine the most promising nodes

2) **Expansion** ; Once a leaf node is reached, Expands the tree by randomly picking a node from a set of possible child nodes

3) **Simulation** ; Monte Carlo simulations (also known as *rollouts*) from newly expanded node → Play out random sequences of actions until reaching a terminal state, yielding an outcome or reward

4) **Backpropagation** ; After a simulation is completed, the result is backpropagated up the tree. The statistics of the visited nodes, such as the number of visits and accumulated rewards, are updated accordingly

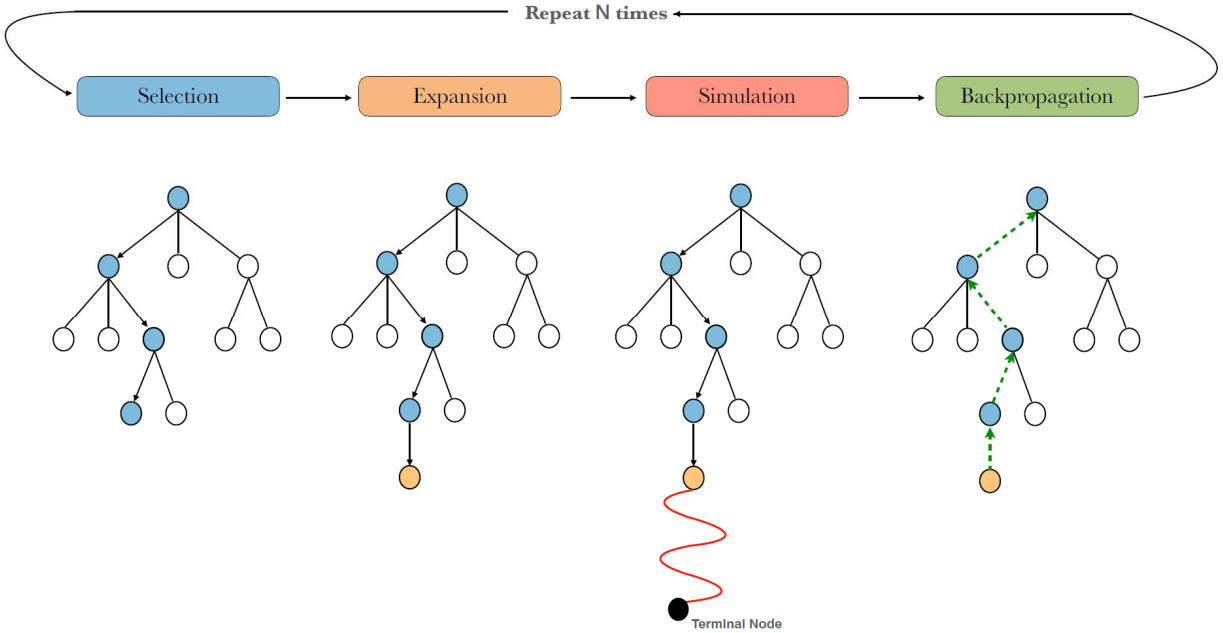


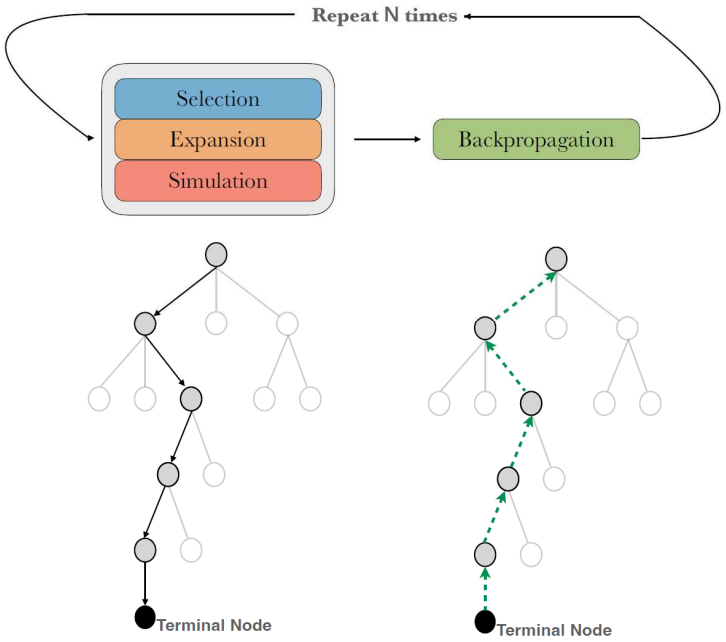
Figure 1: The four major steps of MCTS is depicted in the above figure. These steps are executed sequentially 'N' times

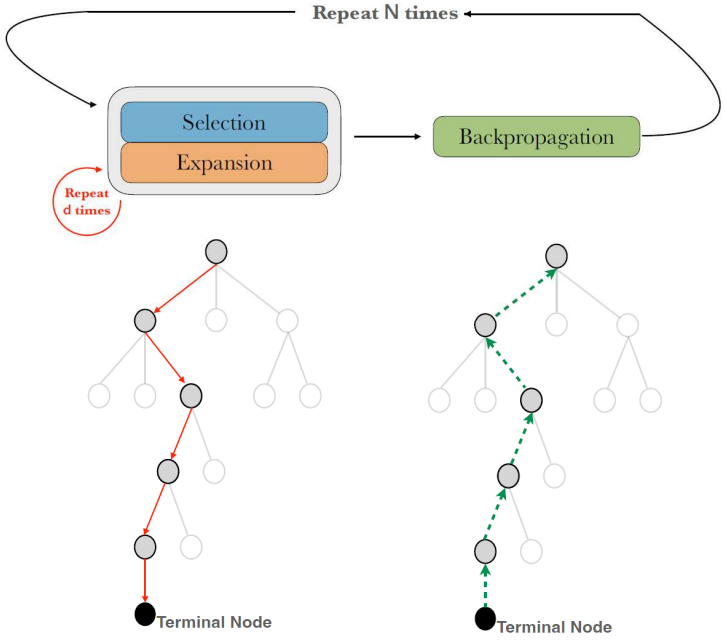
Rollout : 데이터 transition{(s,a,s',r)}을 얻기 위해 Agent와 Environment이 1번(or n번) 상호 작용 하는 것

[Murthy, et al. "REX: Rapid Exploration and eXploitation for AI agents". **ICLR 2024** LLM Agents Workshop.]

Juan Carlos Niebles, Salesforce

REX





REX-UCB : Integration of rewards into the prompt utilizing the **Upper Confidence Bound (UCB)** framework, thereby enabling a systematic exploration and exploitation of the action space

REX-UCL : Systematically influencing the logits of LLMs through what we term **UCL** (**UCB applied for Logits of LLM**), providing enhanced control over the generation of actions.

Figure 2(a) **REX-UCB** ; The Selection, Expansion, and Simulation steps in MCTS is combined to form one single step in REX.

Figure 2(b) **REX-UCL** ; The Selection and Expansion steps are amalgamated into a single step; however, due to the adjustment of logits, all the steps are not simultaneously generated; instead, **each step is generated individually**.

[Murthy, et al. “REX: Rapid Exploration and eXploitation for AI agents”. **ICLR 2024** LLM Agents Workshop.]

Juan Carlos Niebles, Salesforce

REX

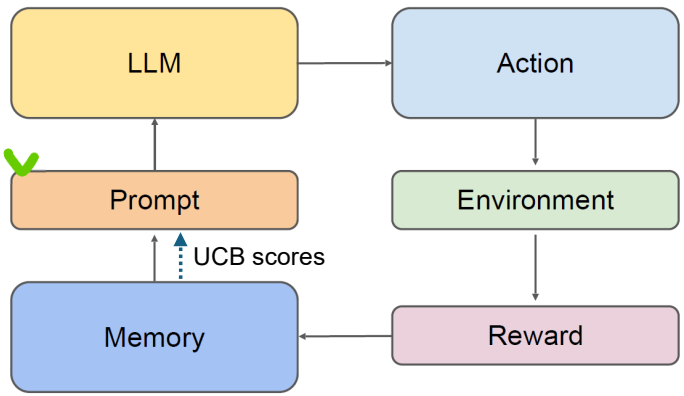


Figure 3(a) **REX-UCB** ; **Action** denotes the solution produced by the **LLM**, while **Environment** represents the context within which the task is defined. The determination and subsequent updating of the **Reward** within the **Memory** (**UCB table**) are contingent upon the task's characteristics. Following this, **the UCB scores are translated into HIGH or LOW values and are subsequently integrated into the Prompt.**

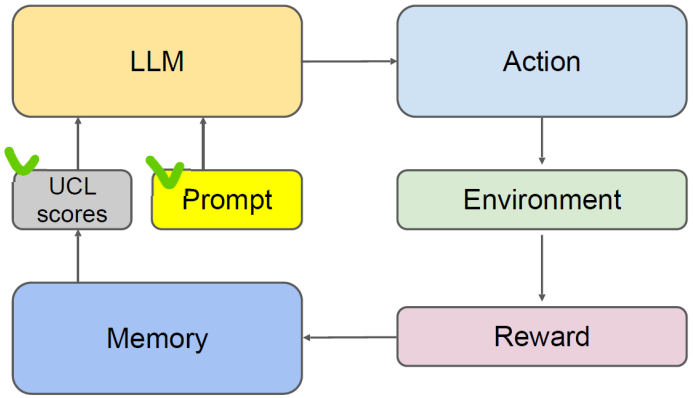


Figure 3(b) **REX-UCL** ; **Action** denotes the solution produced by the **LLM**, while **Environment** represents the context within which the task is defined. The determination and subsequent updating of the **Reward** within the **Memory** (**UCL table**) are contingent upon the task's characteristics. Following this, **the UCL scores are directly provided to LLM to influence the tokens associated with the desired action sequence**

REX-UCB : Integration of rewards into the prompt utilizing the **Upper Confidence Bound (UCB)** framework, thereby enabling a systematic exploration and exploitation of the action space

REX-UCL : Systematically influencing the logits of LLMs through what we term **UCL** (**UCB applied for Logits of LLM**), providing enhanced control over the generation of actions.

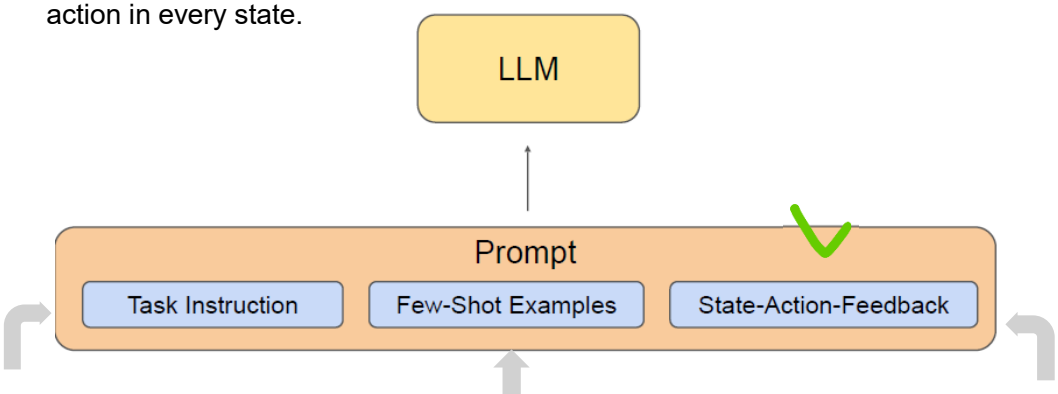
[Murthy, et al. "REX: Rapid Exploration and eXploitation for AI agents". **ICLR 2024** LLM Agents Workshop.]

Juan Carlos Niebles, Salesforce

REX

Prompt Structure for REX-UCB

(1) Task Instruction : The set of specific directions, guidelines, or information provided to LLM to complete a particular task or achieve a specific objective.
(2) Few-Shot examples : The model is expected to generalize from this limited set of examples to perform tasks or make predictions on new, unseen data;
(3) State-Action-Feedback : Expected reward, based on *UCB scores*, for each action in every state.



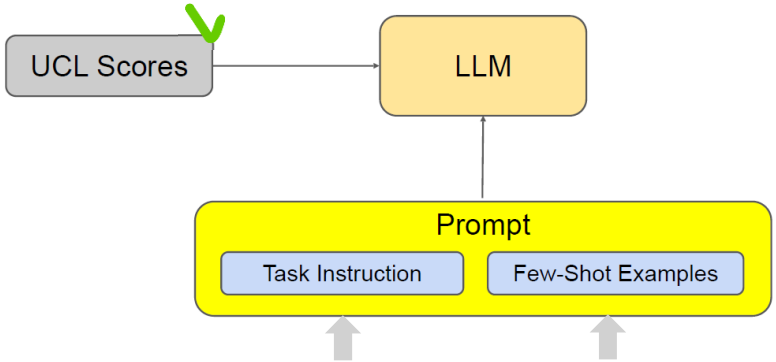
You are a helpful agent capable of solving logical problems. You are required to take a sequence of action to transform Initial Block Configuration to Final Block Configuration [....]
Possible action you can take: [....]
Rules: [...]

Example 1:
Initial Block Configuration: [...]
Final Block Configuration [....]
Actions: [...]
Example 2:
Initial Block Configuration: [...]
Final Block Configuration [....]
Actions: [...]

Actions and Expected Rewards:
Step 1:
- Stack blue on orange has HIGH reward
- Untack Green from Yellow has LOW reward
Step 2:
[...]

Prompt Structure for REX-UCL

The Task Instruction and Few-Shot Examples components are the same as in REX-UCB. State-Action-Feedback component is removed and instead UCL scores are provided to LLM to adjust the logit values.



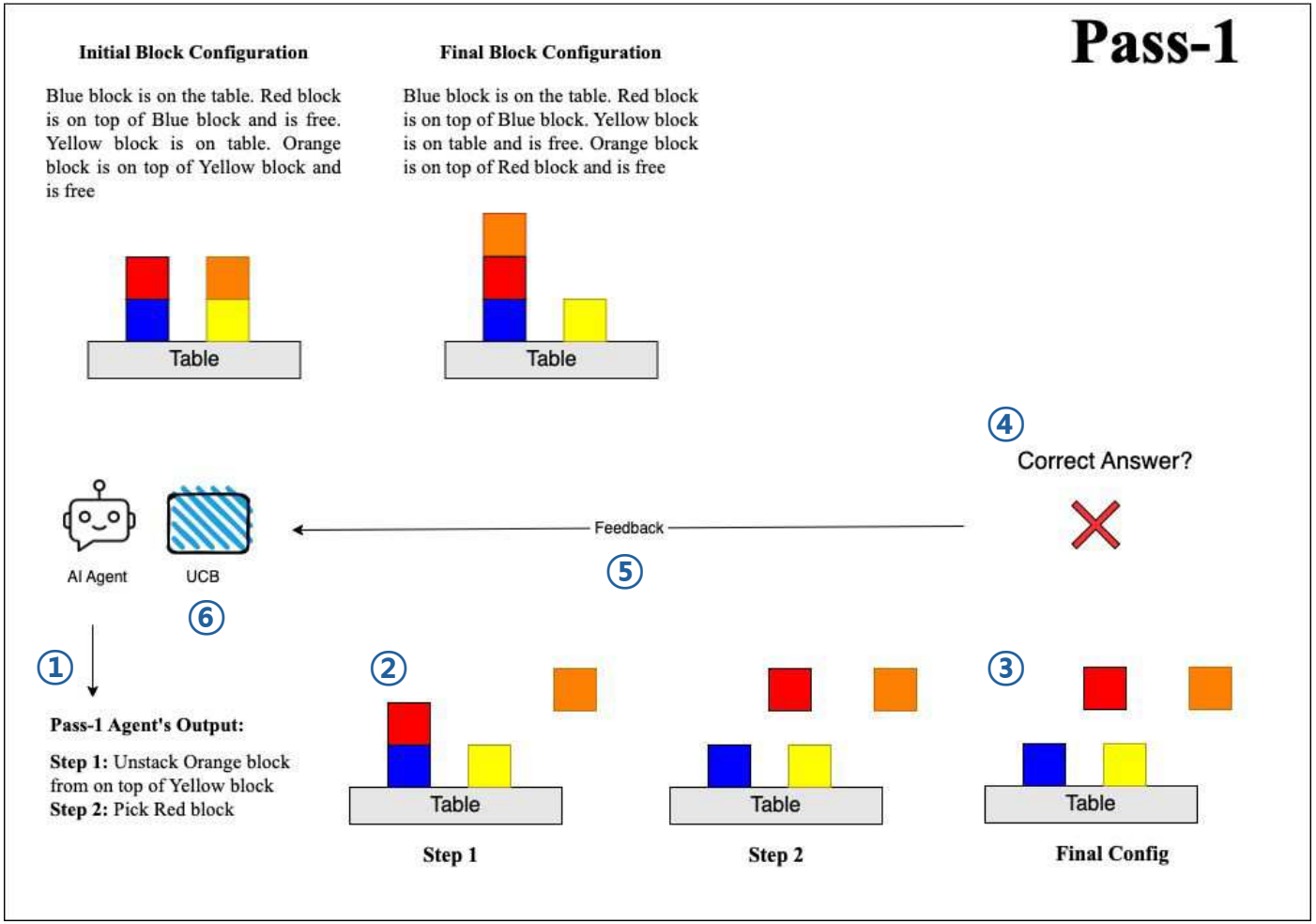
You are a helpful agent capable of solving logical problems. You are required to take a sequence of action to transform Initial Block Configuration to Final Block Configuration [....]
Possible action you can take: [....]
Rules: [...]

Example 1:
Initial Block Configuration: [...]
Final Block Configuration [....]
Actions: [...]
Example 2:
Initial Block Configuration: [...]
Final Block Configuration [....]
Actions: [...]

[Murthy, et al. “REX: Rapid Exploration and eXploitation for AI agents”. **ICLR 2024** LLM Agents Workshop.]

Juan Carlos Niebles, Salesforce

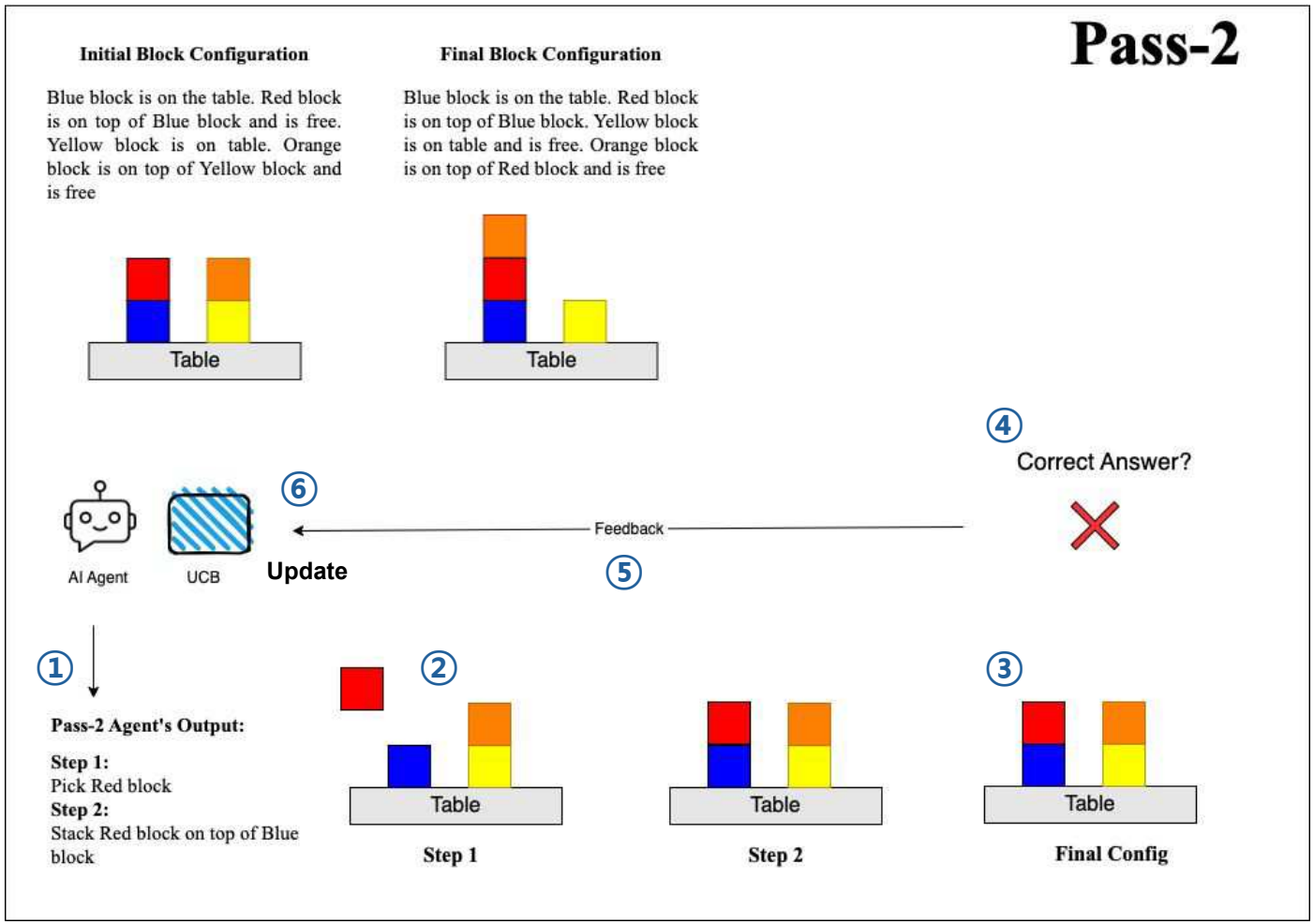
REX Agent Architecture



REX: UCB-CoT

Juan Carlos Niebles, Salesforce

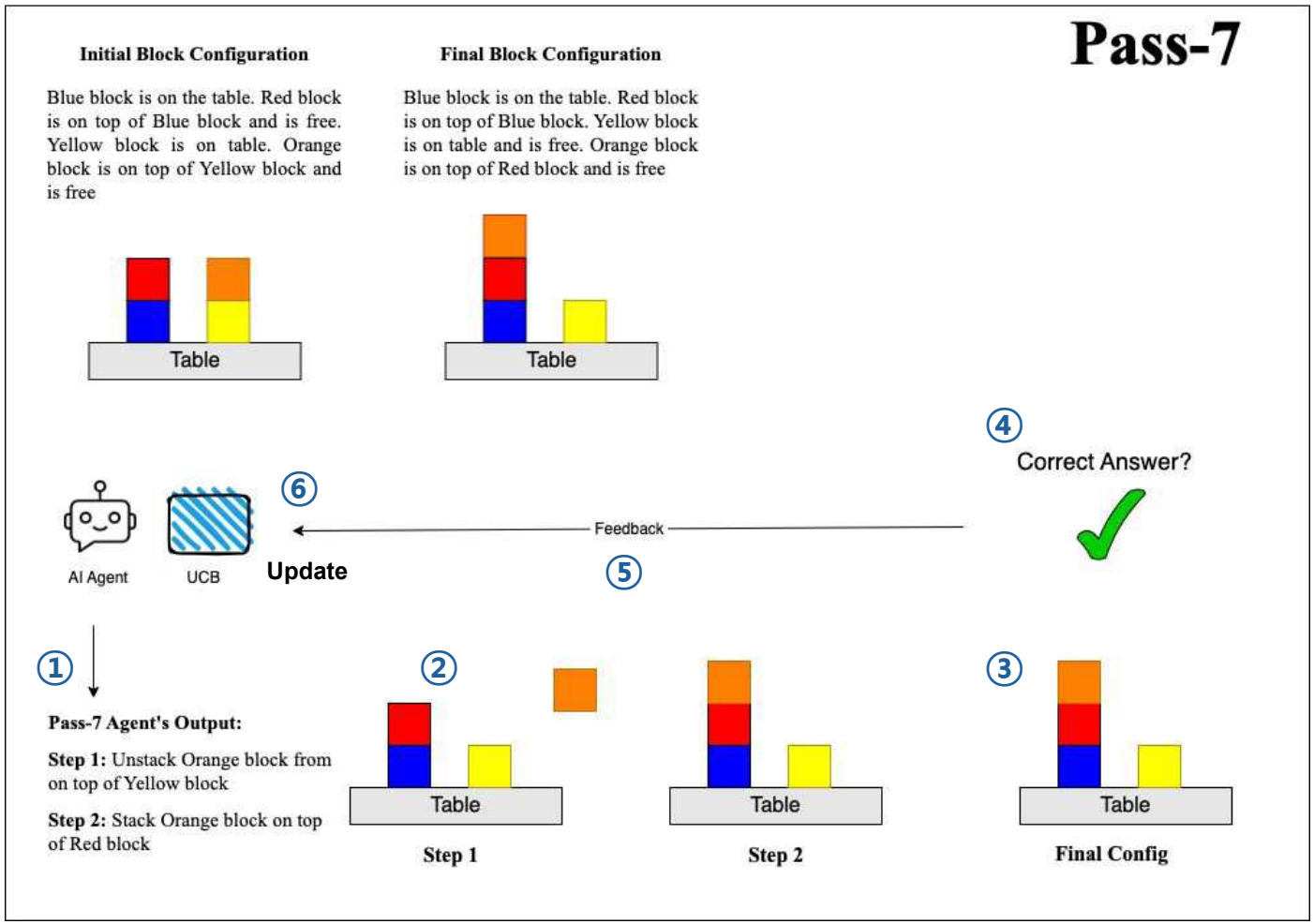
REX Agent Architecture



REX: UCB-CoT

Juan Carlos Niebles, Salesforce

REX Agent Architecture



REX: UCB-CoT

REX Results

Table 1: Accuracy & Time complexity of various models

Architecture	Blocksworld			GSM8K-test	Time complexity
	2-step (size=30)	4-step (size=56)	6-step (size=114)	2-to-8-steps (size=1319)	
CoT	40%	17.85%	8.77%	80.81%	n
Reflexion (+ CoT)	41.67%	<u>41.96%</u>	**29.82%**	<u>88.85%</u>	$3*n$
RAP	-	-	-	-	$n*m*d$
REX-$\mathcal{R}$	53.33%	37.5%	14.91%	81.34%	n
REX-UCB	**80%**	39.28%	<u>25.43%</u>	82.03%	n
REX-UCL	<u>60%</u>	**44.64%**	20.17%	**90.44%**	$n*d$

- **Blocksworld** dataset (Valmeekam et al., 2023) represents a planning problem that involves the arrangement of blocks with varying colors in a predetermined configuration
- **GSM8K** dataset (Cobbe et al., 2021) comprises a collection of 8.5k grade school math word problems of exceptional quality

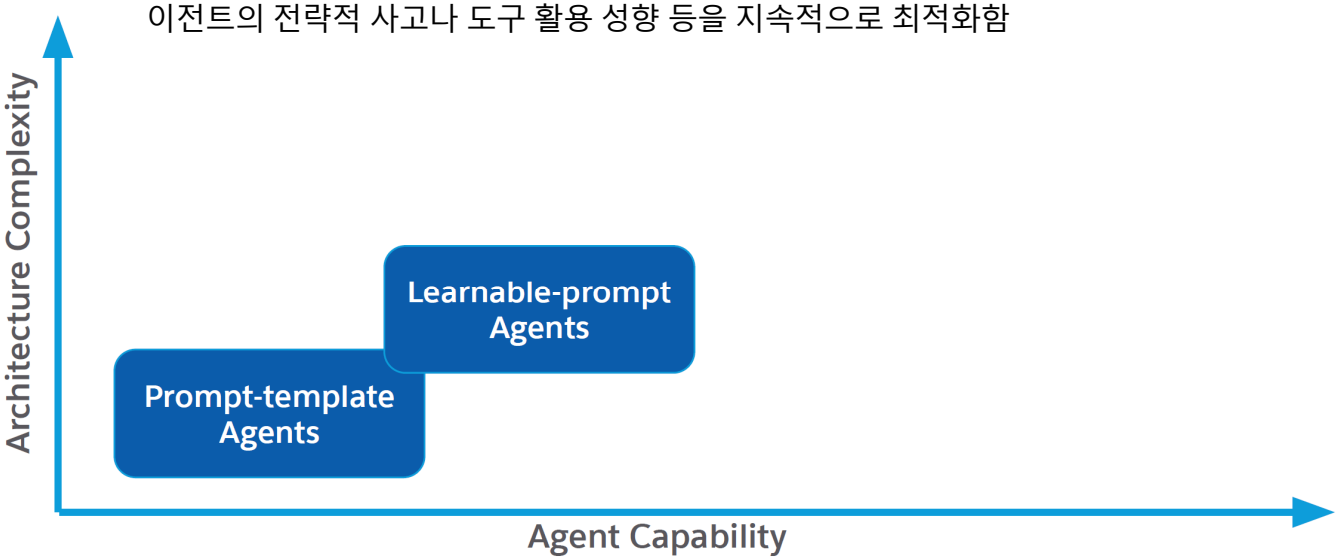
- REX Limitations
  - Prompt Complexity 증가: REX는 Reward 및 UCB score를 Prompt에 통합하므로, Prompt의 Complexity가 증가할 수 있음
  - Reward 설계의 어려움: REX의 성능은 Reward 설계에 크게 의존하므로, 적절한 Reward function을 설계하는 것이 중요함

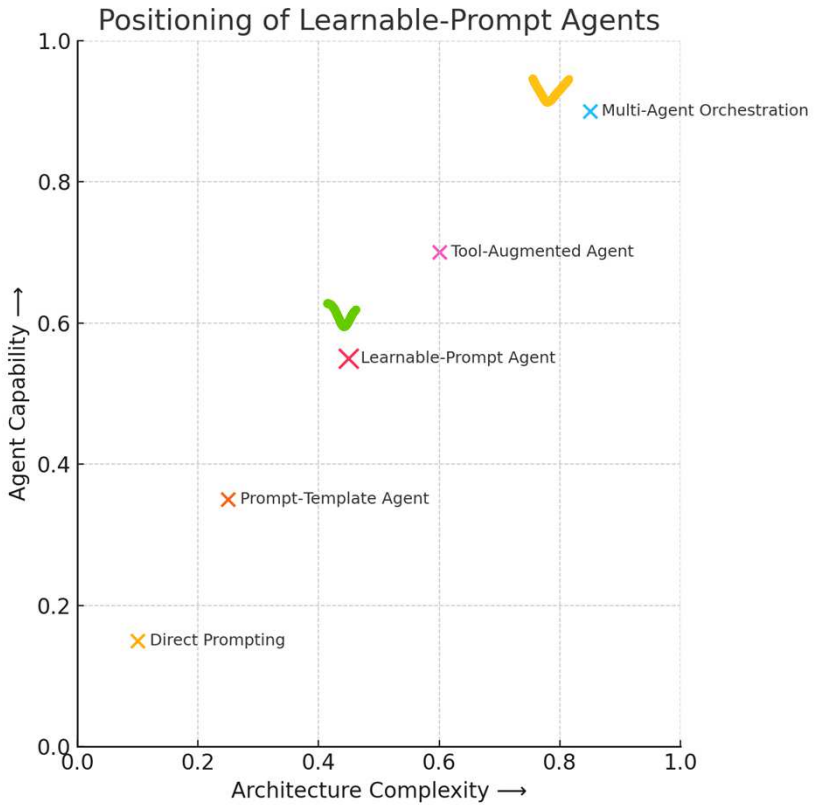
Juan Carlos Niebles, Salesforce

Language-based AI Agents

➢ **Learnable-Prompt Agent**

- 고정된 프롬프트 템플릿 대신, 학습 가능한 프롬프트(reinforced, soft prompts, 또는 meta-prompt tuning)를 사용하여 에이전트의 행동을 동적으로 조정
- 기존 static prompt-template과 달리, 프롬프트 자체가 데이터 기반으로 조정되며, 에이전트의 전략적 사고나 도구 활용 성향 등을 지속적으로 최적화함





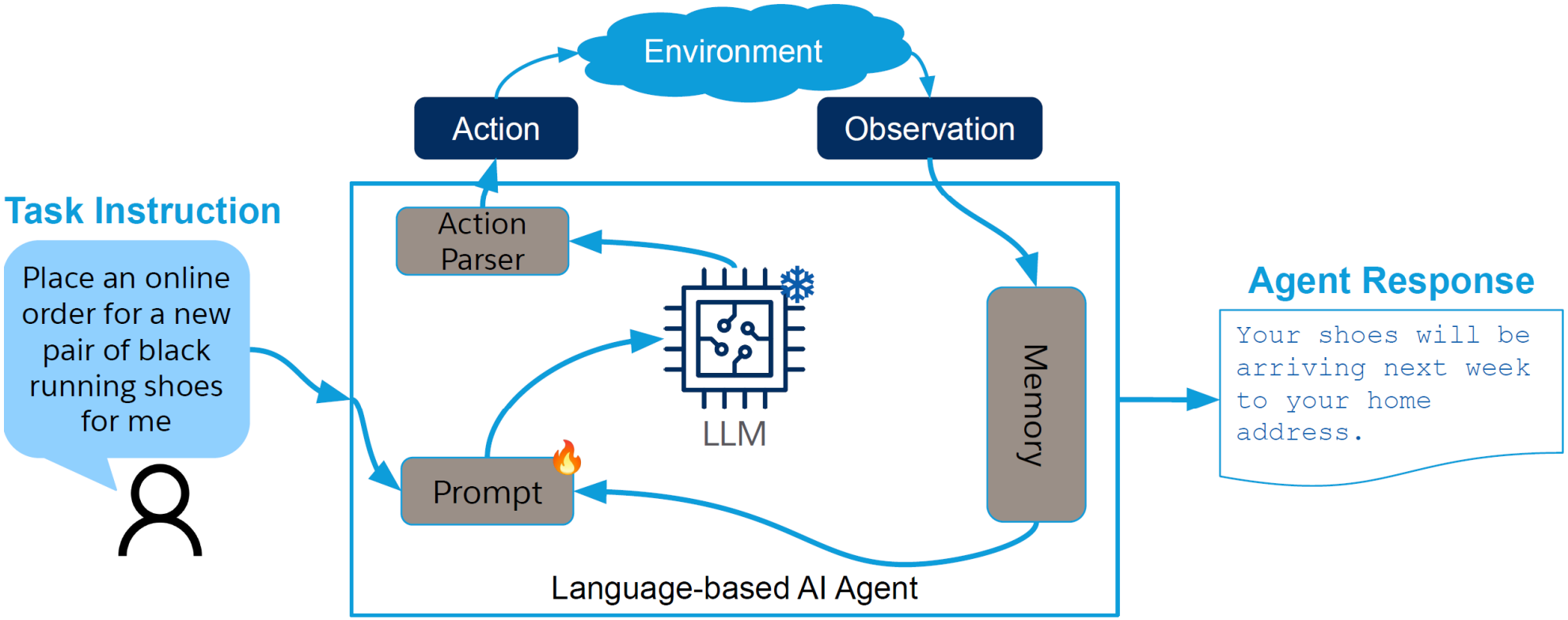
- Architecture Complexity : 중간 이상 (≈ 0.45)
  - 학습 가능한 프롬프트 구성 모듈 필요
  - 프롬프트 업데이트를 위한 추가 컨트롤 구조 존재
  - 외부 환경의 feedback loop가 통합됨
- Agent Capability : 중간 이상 (≈ 0.55)
  - 고정 템플릿보다 더 복잡한 문제 대응 가능
  - 다양한 Task에서 동적으로 행동 조정 가능
  - reasoning + reward alignment에 유리

[☞ ChatGPT o1]

Agent Architectures

Juan Carlos Niebles, Salesforce

Learnable-prompt Agents

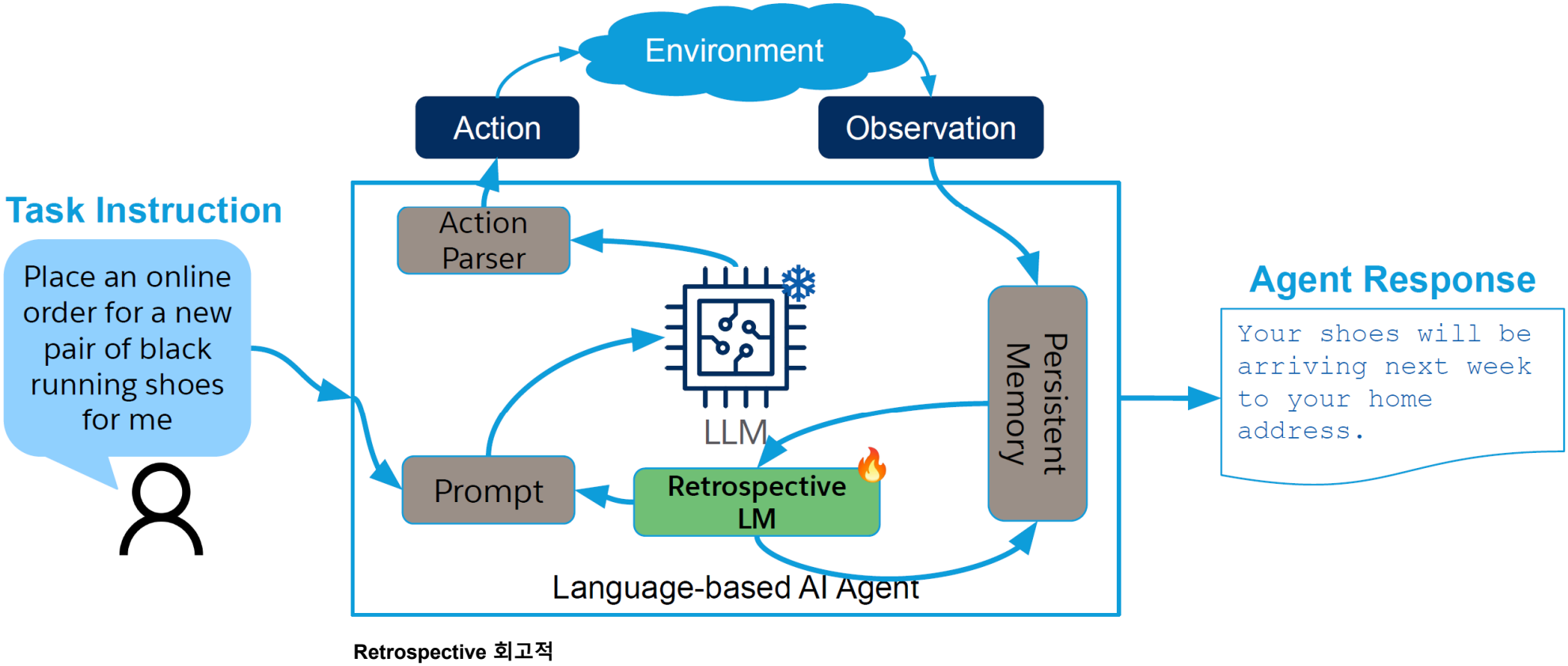


- 🧠 **학습 기반 Prompt 최적화** : Prompt를 gradient 기반으로 학습하거나 RL 기반 보상을 통해 수정
- 🎯 **Task 특화 Prompt 구성 가능** : 도메인별 응답 정확도 향상
- 🔁 **자기 반영(Self-Reflection) 기반 개선** : Agent가 이전 실행을 분석해 Prompt를 업데이트
- 📦 **Pretrained LLM과 호환** : Prompt tuning만으로 zero-shot/low-resource 설정에서 성능 향상 가능

Juan Carlos Niebles, Salesforce

Retroformer

https://github.com/weirayao/Retroformer



Retrospective 회고적

[Yao et al. Retroformer: Retrospective Large Language Agents with Policy Gradient Optimization. **ICLR 2024**]

Juan Carlos Niebles, Salesforce

Retroformer

https://github.com/weirayao/Retroformer

➢ **Policy gradient optimization 활용하여, Agent가 이전 실패 경험을 반영하여 프롬프트를 개선**

- Most existing language agents; Not optimized using environment-specific rewards.
- Although some agents enable iterative refinement through verbal feedback, they do not reason and plan in ways that are compatible with gradient-based learning from rewards.

- **Retroformer;**
- Reinforce large language agents by learning a **retrospective model**, which **automatically tunes the language agent prompts from environment feedback through *policy gradient***.
- Learns from rewards across multiple environments and tasks, for fine-tuning a pre-trained language model which refines the language agent prompt by summarizing the root cause of prior failed attempts and proposing action plans.

- 2 Main Components of Retroformer

- **Actor LLM**, M_a; Generates reasoning thoughts and actions
  - ✓ Pre-trained LLM으로, 주어진 Prompt에 따라 Action a_t을 생성함. 이 모델은 고정되어 있고, 직접적인 파라미터 업데이트 없이 사용
  - ✓ Use ReAct as Actor Prompt

$$a_{k,i,t} = M_a\left([s_{k,i,\tau}, a_{k,i,\tau}, r_{k,i,\tau}]_{\tau=1}^{t-1}, s_{k,i,t}\right).$$

- **Retrospective LLM**, M_r; Generates verbal reinforcement cues to assist the actor in self-improvement by refining the actor prompt with reflection responses.
  - ✓ 작은 규모의 언어 모델로, Agent의 이전 실패 경험을 요약하고 분석하여, 향후 행동 계획을 제안함. 이 모델은 환경으로부터 받은 보상 신호를 기반으로 Policy gradient 방법을 통해 학습됨.
  - ✓ Self-reflection feedback $y_{k,i}$

$$y_{k,i} = M_r(\underbrace{[s_{k,i,\tau}, a_{k,i,\tau}, r_{k,i,\tau}]_{\tau=1}^{T}, G_{k,i}}_{\text{Reflection prompt } x_{k,i}}).$$

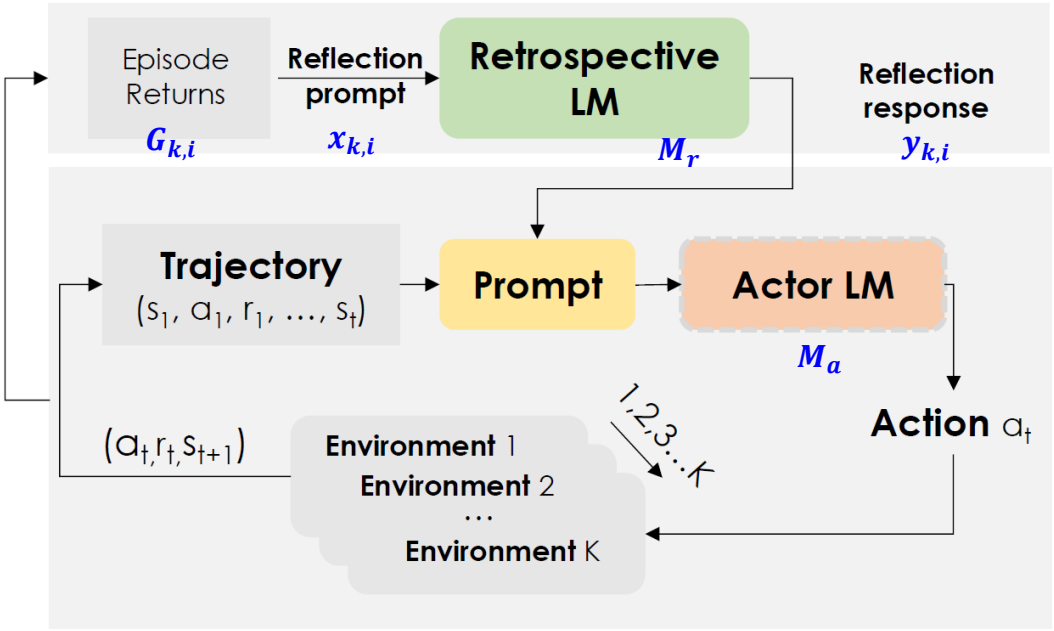
Episode return

[Yao et al. Retroformer: Retrospective Large Language Agents with Policy Gradient Optimization. **ICLR 2024**]

Juan Carlos Niebles, Salesforce

Retroformer

https://github.com/weirayao/Retroformer



(a) Retrospective agent

Figure 2: Framework overview. (a) The retrospective agent system (Sec. 4.1) contains two LLMs communicating to refine agent prompts with environment feedback.

- **Actor model** is regarded as **a frozen LLM**, such as GPT, with inaccessible model parameters.
- In this scenario, **the most direct approach to enhancing actor performance in a given environment** is by **refining the actor LM's prompt**.
- Consequently, **retrospective model**, *a smaller local language model*, **refines the actor's prompt** by **incorporating a concise summary of errors and valuable insights from failed attempts**.

- Aim to **optimize the retrospective model using environment reward**.
- The desired behavior of M_r is to improve the actor model M_a in next attempt.
- Hence, the difference in episode returns between two consecutive trials serves as a reward signal for fine-tuning the retrospective model with reinforcement learning.

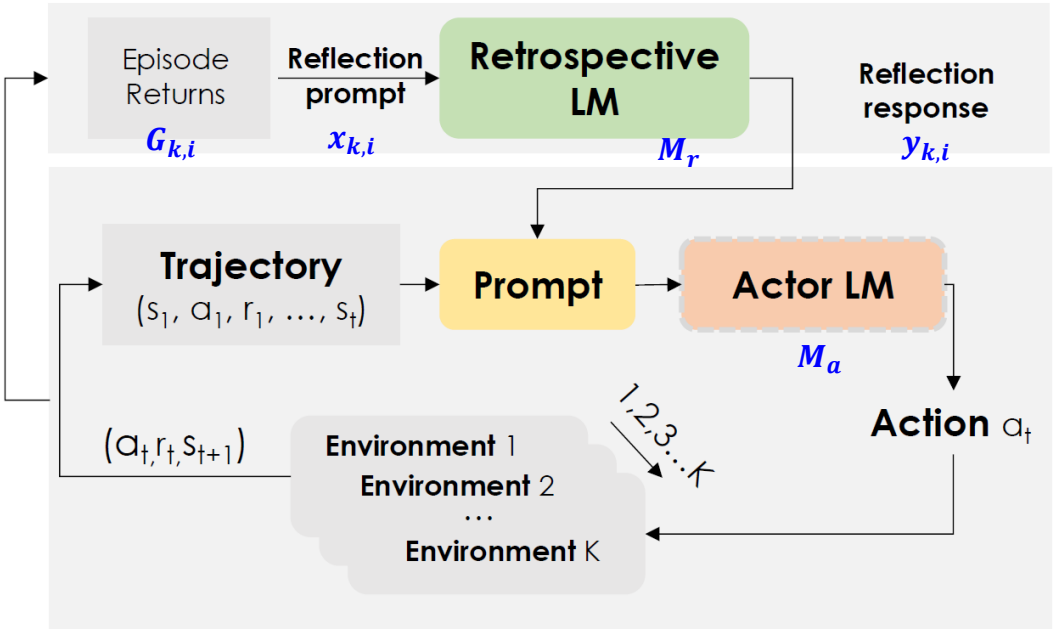
- Let us assume a **reflection prompt** $x_{k,i}$ and the corresponding **episode return** $G_{k,i}$, and the **retrospective model** M_r generates the **response** $y_{k,i}$ that summarizes the mistakes in i, which results in the return $G_{k,i+1}$ in the next attempt $i+1$.

[Yao et al. Retroformer: Retrospective Large Language Agents with Policy Gradient Optimization. **ICLR 2024**]

Juan Carlos Niebles, Salesforce

Retroformer

https://github.com/weirayao/Retroformer



(a) Retrospective agent

Figure 2: Framework overview. (a) The retrospective agent system (Sec. 4.1) contains two LLMs communicating to refine agent prompts with environment feedback.

➢ **Policy Gradient Optimization**

- **Instruction and Response Generation**

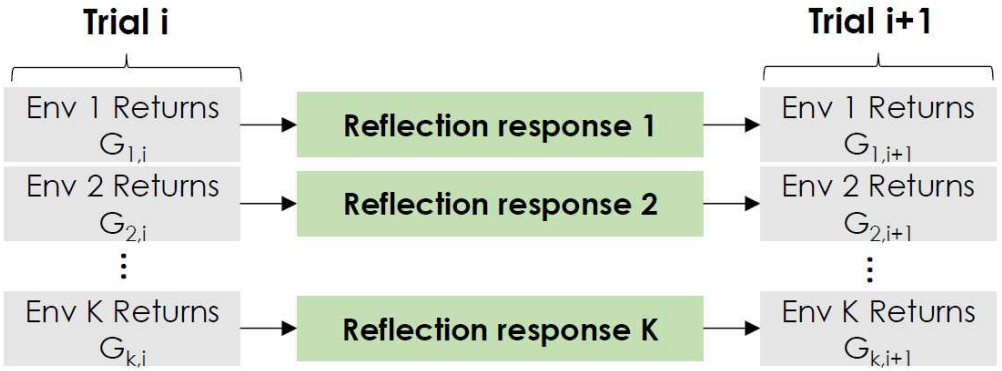
- **Retrospective model** M_r generates a pair of (*instruction*, *response*) at the end of each episode i in the environment k.
- In the episode i, the **actor** produces a **trajectory** τ_i by interacting with the environment. The **reward function** then produces a **score** r_i. At the end of the episode, to produce verbal feedback for refining the actor prompt, M_r takes the set of $\{\tau_i, r_i\}$ as the instruction $x_{k,i}$ and is prompted to produce a reflection response $y_{k,i}$.
- All instruction-response pairs $(x_{k,i}, y_{k,i})$ across tasks and trials are stored to a local dataset DRL, "**replay buffer**", for fine-tuning the M_r.

[Yao et al. Retroformer: Retrospective Large Language Agents with Policy Gradient Optimization. **ICLR 2024**]

Juan Carlos Niebles, Salesforce

Retroformer

https://github.com/weirayao/Retroformer



(b) Ratings for reflection responses

Figure 2: Framework overview (b) **The retrospective LM is fine-tuned with response ratings using proximal policy optimization** (Sec. 4.2).

➢ Policy Gradient Optimization

- **Response Rating**

- Because the actor is a frozen LM and the temperature is low as default, the injected randomness that leads to differences in returns $\Delta G_{k,i} = G_{k,i+1} - G_{k,i}$ are mostly from the reflection responses $y_{k,i}$.
  - **Positive $\Delta G_{k,i}$** indicates **better responses** that help the actor learn from prior errors, and hence should be rated with higher scores;
  - **Negative or zero $\Delta G_{k,i}$** indicates **worse responses** that needs to be avoided and hence should be rated with lower scores.
- Therefore, we approximate **the rating score of a reflection instruction-response pair** $(x_{k,j}, y_{k,j})$ as:

$$r(x_{k,i}, y_{k,i}) \triangleq G_{k,i+1} - G_{k,i}.$$

- **Proximal Policy Optimization** (뒷장)

[Yao et al. Retroformer: Retrospective Large Language Agents with Policy Gradient Optimization. **ICLR 2024**]

Juan Carlos Niebles, Salesforce

Retroformer: Policy Gradient Optimization

- Use **Proximal Policy Optimization (PPO)** to fine-tune the retrospective LM

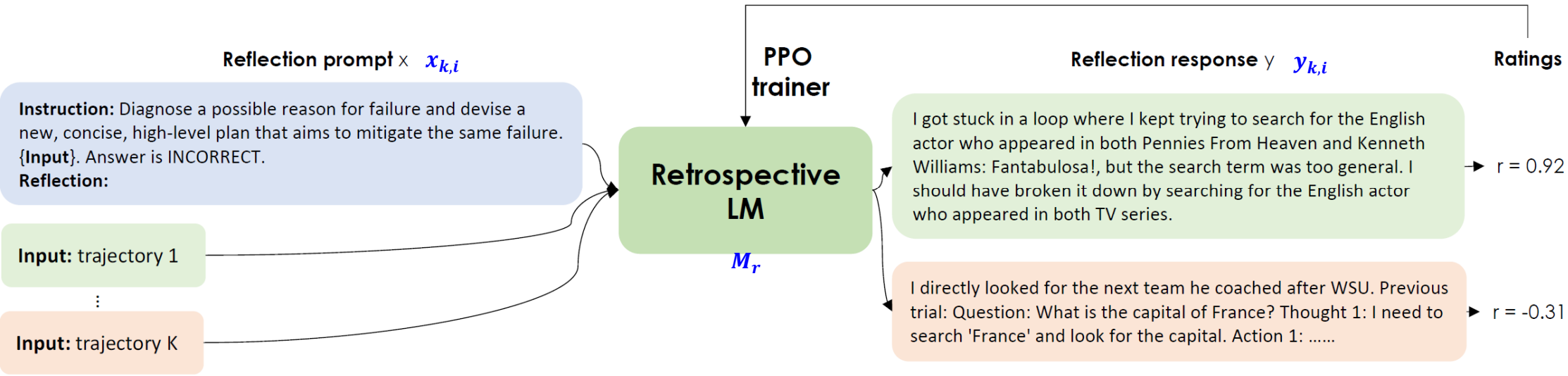


Figure 3: Policy gradient optimization of retrospective LM using RLHF training pipeline.

- **Retrospective LM** is **fine-tuned** with the **response ratings** following the **RLHF training** procedures with **proximal policy optimization (PPO)**:

$$\mathcal{L}_{\text{PPO}} = \mathbb{E}_{x\sim D_{\text{RL}}}\mathbb{E}_{y\sim \text{LLM}_{\phi}^{\text{RL}}(x)}\left[r_\theta(x,y) - \beta \log \frac{\text{LLM}_{\phi}^{\text{RL}}(y|x)}{\text{LLM}^{\text{Ref}}(y|x)}\right]$$

Policy object function

- (x, y) : Sampled from the replay buffer
- $r_\theta(x, y)$: the Defined reward model
- Second term in this objective : KL divergence to make sure that the fine-tuned model LLM^{RL} does not stray too far from the frozen reference model LLM^{Ref}.

Retroformer

https://github.com/weirayao/Retroform

Our reinforced response prompts the agent to focus on spinoff series and asks the agent to find the answer in the previous actions and observations with the search results of “Tara Strong”. Retroformer presents better credit assignment and root cause analysis abilities, and has the power to generate actionable insights.

Question	Prior failed trial	Reflection (Green: Retroformer; Brown: Frozen LM)
Lollipop Chainsaw featured Juliet Starling, who was voiced by a Canadian-American actress who has done voice roles for what Teen Titans **spinoff series**?	**Thought 1:** I need to search Juliet Starling and find the Canadian-American actress who voiced her, then find the Teen Titans spinoff series she has done voice roles for. **Action 1:** Search[Juliet Starling] **Observation 1:** Could not find [Juliet Starling]. … **Thought 2:** Juliet Starling is a character from Lollipop Chainsaw, so I can search Lollipop Chainsaw. **Action 2:** Search[Lollipop Chainsaw] **Observation 2:** Lollipop Chainsaw is a … **Thought 3:** Juliet Starling is voiced by Tara Strong, so I need to search Tara Strong and find the Teen Titans spinoff series she has done voice roles for. **Action 3:** Search[Tara Strong] **Observation 3:** Tara Lyn Strong… **Thought 4:** Tara Strong has voiced characters in Teen Titans and Teen Titans Go!, so the answer is Teen Titans and Teen Titans Go!. **Action 4:** Finish[Teen Titans and Teen Titans Go!]	I failed to find the Teen Titans spinoff series that Tara Strong has done voice roles for. Instead, I found Lollipop Chainsaw, a video game that Juliet Starling appeared in. I should have searched Tara Strong and Teen Titans or Teen Titans Go! specifically in my previous action to find the correct answer My thought process was strong t should have searched for Lollipop Chainsaw first and looked up the Canadian-American actress who voiced Juliet Starling afterwards. I also should have looked up Tara Strong's filmography and searched for any voice roles she did specifically for Teen Titans or Teen Titans Go!.Next trial:Question: The novel was adapted into a film by a director known for his work on another iconic 1980s franchise. What novel is this film based on?Thought 1:

② The self-reflection from the frozen model reiterated the prior action sequences that led to failure as the steps that the agent should have done, which prompts the agent to repeat these steps in the next attempt, resulting in an infinite loop.

① The agent failed in the last attempt : the correct answer includes only “**Teen Titans Go**”;

The agent forgot its original goal during a chain of lengthy interactions.

Figure 5: Response refinement from the reinforced retrospective model. Note that the lengthy observation step in the prior failed trial column is abbreviated for better presentation purposes
→ **It illustrates how the uninformative self-reflections from a frozen LLM are tackled by RL**.

Juan Carlos Niebles, Salesforce

Retroformer: HotPotQA Result

Question Answering Task

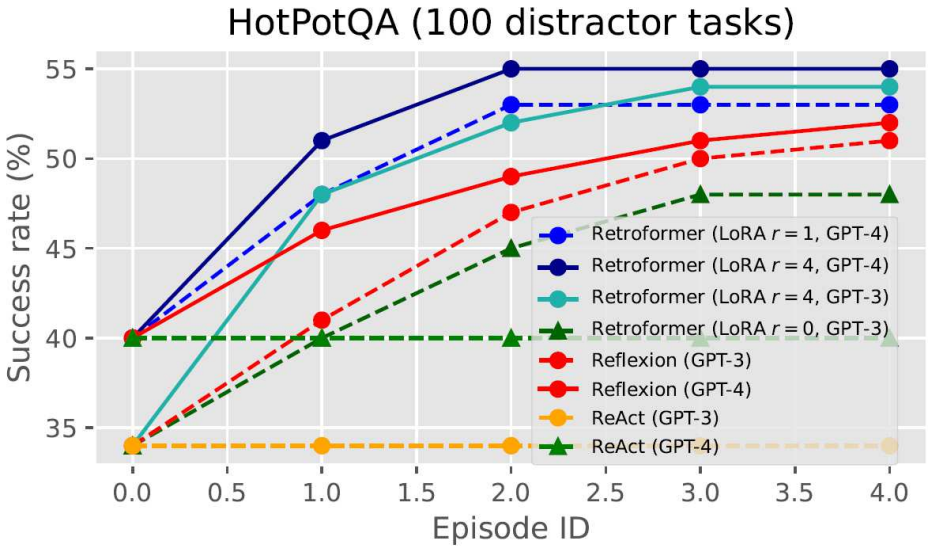


Figure 4: **Retroformer** shows faster and consistent performance improvement of success rate.

Task Description

- Solve a question answering task by searching in Wikipedia pages.
- At each step, the agent is asked to choose from three action APIs:
  - ✓ Search[entity]
  - ✓ Lookup[keyword]
  - ✓ Finish[answer]
- Evaluation metric (success rate) measures if the answer and ground truth matches exactly or not, over all tasks.

[Yao et al. Retroformer: Retrospective Large Language Agents with Policy Gradient Optimization. **ICLR 2024**]

Juan Carlos Niebles, Salesforce

Retroformer: Alfworld & Webshop Results

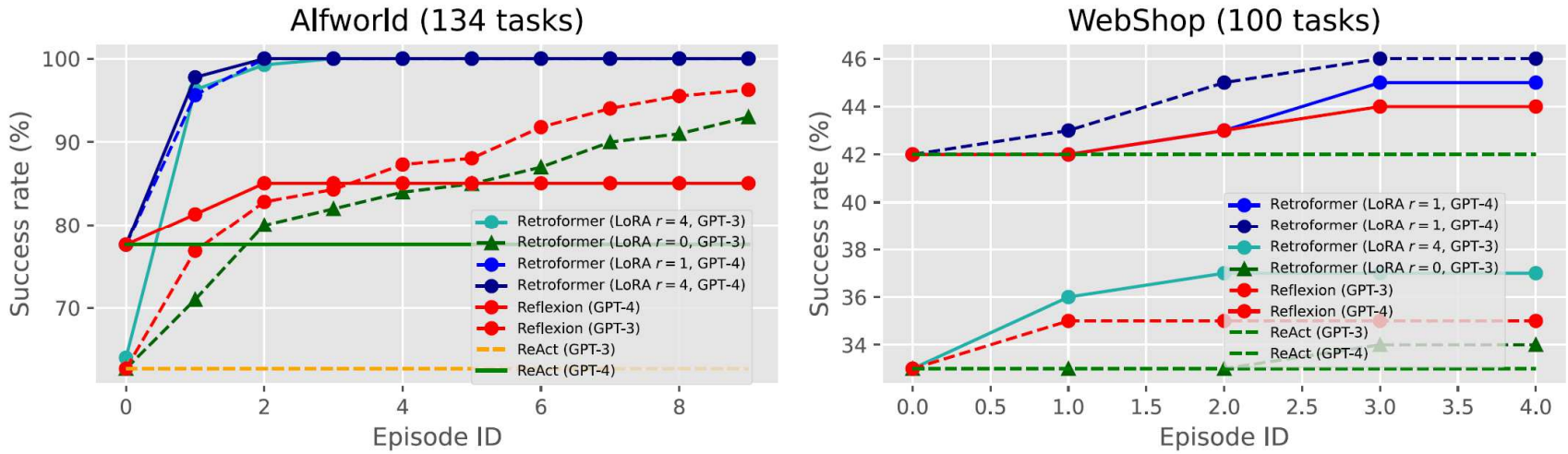


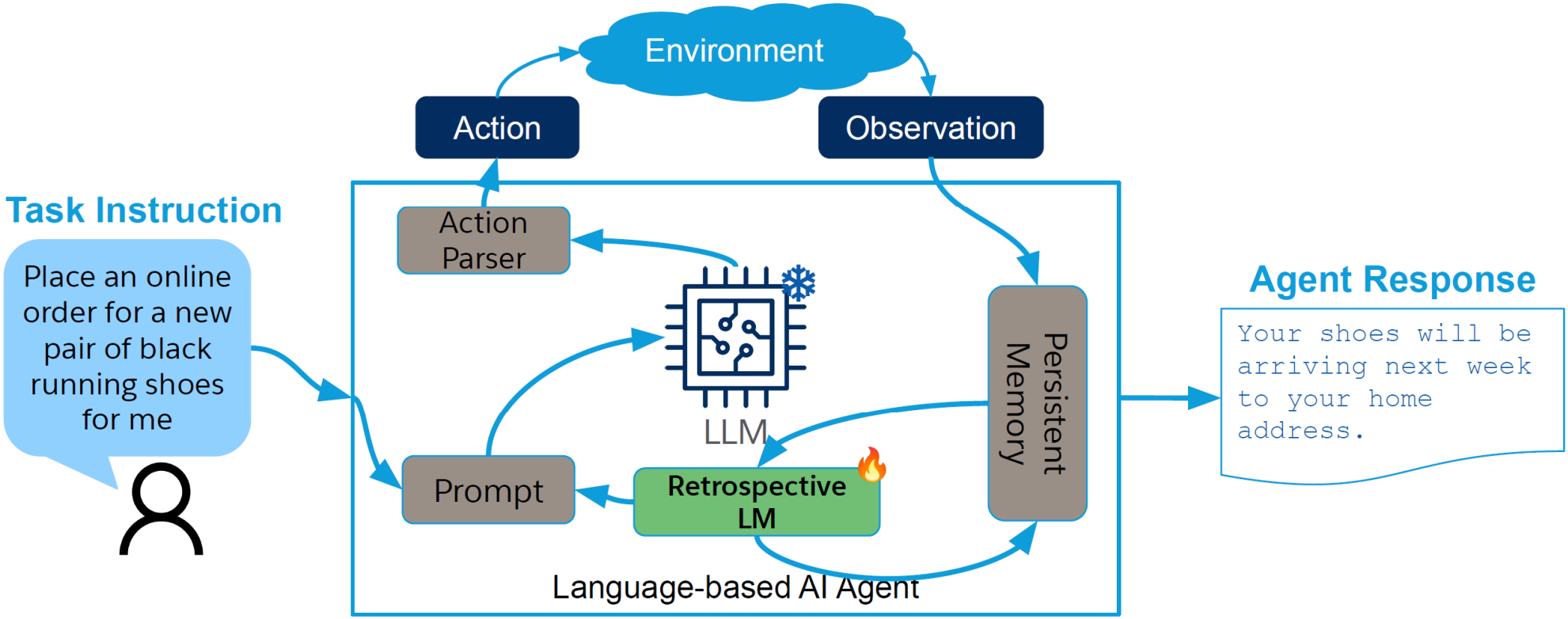
Figure 6: Comparisons of **Retroformer** against baselines in (a) AlfWorld and (b) WebShop environments under different base Actor LLM and LoRA rank r = 1, 4.

[Yao et al. Retroformer: Retrospective Large Language Agents with Policy Gradient Optimization. **ICLR 2024**]

Juan Carlos Niebles, Salesforce

Retroformer takeaways: Refine prompts, decompose

https://github.com/weirayao/Retroformer



[Yao et al. Retroformer: Retrospective Large Language Agents with Policy Gradient Optimization. **ICLR 2024**]

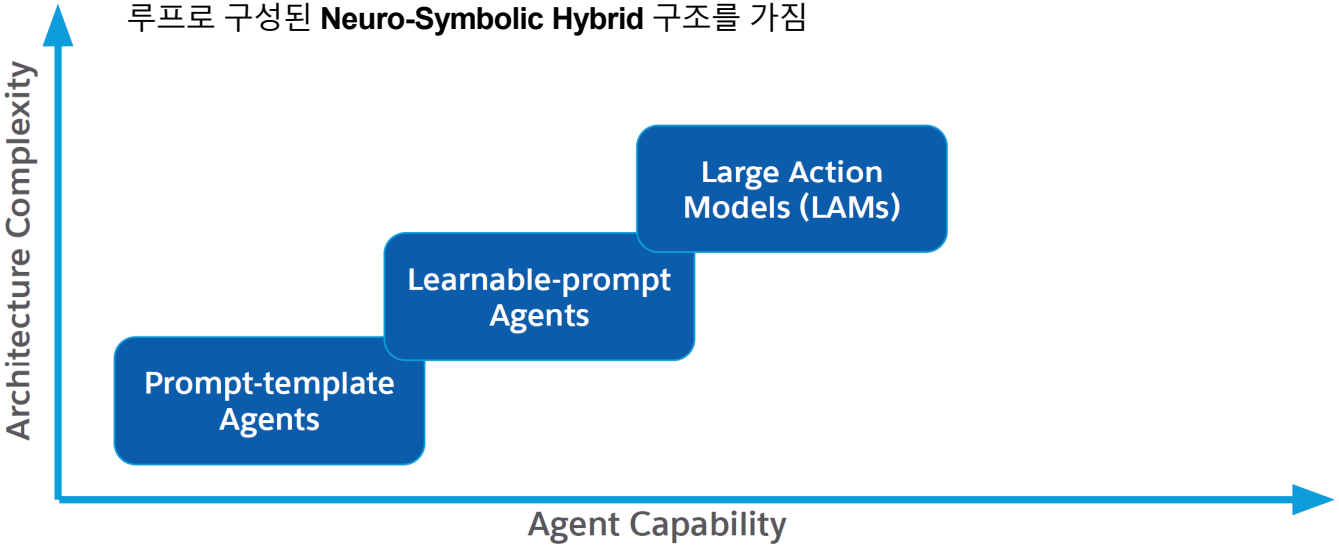
Agent Architectures

Juan Carlos Niebles, Salesforce

Language-based AI Agents

➢ **Large Action Models (LAMs)**

- LLM-기반 Agent가 "**Language(언어 이해·생성)**"을 넘어 "**Action(도구 호출·UI 조작·API 연동)**"까지 담당할 수 있도록 **Action Space**을 직접 출력하도록 학습된 거대 모델 (Ex; Google RT2, Rabbit R1, Salesforce xLAM, Adept ACT-1 등)
- 최신 LAM들은 행동 계층(hierarchy), 계획 엔진(planner), 실행 모듈(executor), 지속 학습 루프로 구성된 **Neuro-Symbolic Hybrid** 구조를 가짐



- Architecture Complexity : High
  - 상대적으로 높은 복잡도; ① 도구 메타데이터·권한·샌드박스 관리, ② 상태·메모리·로그 저장소 필요, ③ 실패 시 롤백·재시도 로직, ④ 보안·감사 체계 탑재
- Agent Capability : High
  - LAMs는 높은 수준의 작업 자동화 가능 (멀티스텝 계획, 복합 작업 수행)
  - 단일 명령으로 UI 네비게이션부터 API 통합까지 처리

LAM의 핵심 아키텍처

Multimodal Encoder
자연어·음성·이미지·UI 스냅숏 등 인코딩

Observation

Intent Parser (& Context Builder)
사용자 목표·제약·환경 상태 추출
메모리·DB·도메인 지식 결합

Goal Spec

Planner / Task Decomposer
ReAct·Tree-of-Thought·TOT-Graph 등 추론
하위 작업(steps)·병렬 작업(branches) 생성
각 Step별 필요한 Tool/API 지정

Action Plan

Tool Selector & Policy Head
OpenAPI/JSON-schema 기반 "Function-calling" 형식화
권한·비용·안전 필터링

Structured Call

Action Executor / Sandbox
코드 실행, HTTP API, UI RPA, 로봇 제어 등
Sandbox·시뮬레이터·로봇 미들웨어 연결
실행 로그·Trace 기록

Result / Error

Observer / Critic (Feedback Loop)
성공 여부·품질·안전성 평가
실패 시 롤백·재시도·휴먼-인-더-루프 요청
Memory Update

Update state

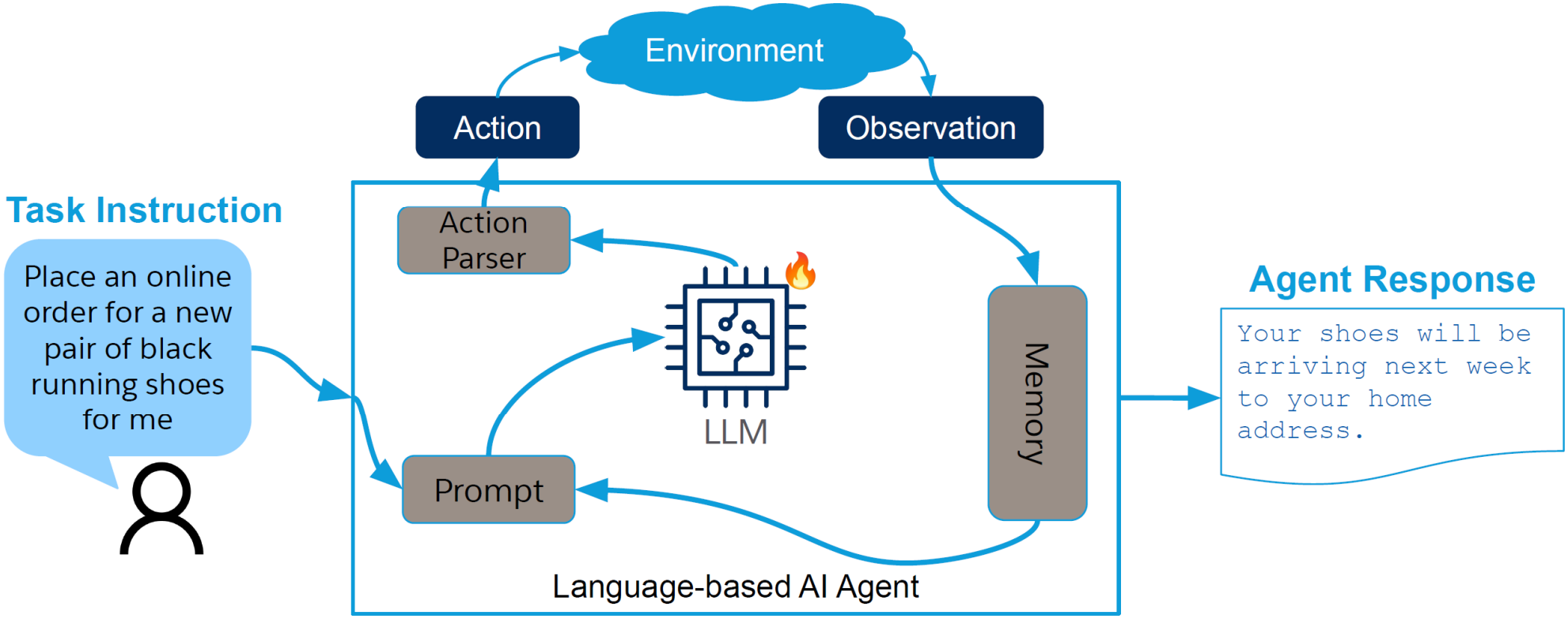
Loop

Agent Architectures

Juan Carlos Niebles, Salesforce

Large Action Models (LAMs)

https://github.com/SalesforceAIResearch/xLAM



[Zhang el al. The AgentOhana: Designing Unified Data and Training Pipeline for Effective Agent Learning. **ICLR 2024** Workshop on LLM Agents]

Agent Architectures

xLAM (eXtended LAM)

https://github.com/SalesforceAIResearch/xLAM

➢ **xLAM** : A series of large action models designed for empowering AI agent system.

- xLAM은 사용자 의도를 실행 가능한 Action으로 변환하여 실제 세계와 상호 작용하는 데 특화
- **Function-Calling** 능력; 사용자의 요청에 따라 외부 시스템과 직접 통신하고, 특정 작업을 수행**, 사용자가 제공한 함수 또는 API 목록 내에서 주어진 상황과 목표에 가장 적합한 함수를 스스로 판단하여 선택하고, 필요한 인수를 결정하며, 선택된 함수의 실행 결과를 얻음**
- 단순히 명령 실행하는 것을 넘어, 요청의 **맥락을 깊이 이해**하고 **정보에 기반한 결정**을 내릴 수 있는 **추론 능력** 내장

- xLAM **aggregates agent trajectories from distinct environments**, spanning a wide array of scenarios. **Standardizes and unifies these trajectories into a consistent format**, **streamlining the creation of a generic data loader optimized for agent training**.
- Leveraging the data unification, our training pipeline maintains equilibrium across different data sources and preserves independent randomness across devices during dataset partitioning and model training.

Model	Base Model	# Total Params	Context Length	Category
xLAM-1b-fc-r	DeepSeek-Coder-1b	1.35B	16k	Function-calling
xLAM-7b-fc-r	DeepSeek-Coder-7b	6.91B	4k	Function-calling
xLAM-7b-r	Mistral-7b	7.24B	32k	General
xLAM-8x7b-r	Mistral-8x7b	46.7B	32k	General
xLAM-8x22b-r	Mistral-8x22b	141B	64k	General

- xLAM series includes 5 models with both dense and mixture-of-expert architectures, ranging from 1B to 8x22B parameters, trained using a **scalable**, **flexible pipeline** that unifies, augments, and synthesizes diverse datasets to enhance AI agents' generalizability and performance across varied environments.
- Securing the 1st position on the **Berkeley Function-Calling Leaderboard**, outperforming GPT-4, Claude-3, and many other models in terms of tool use.

[Zhang el al. xLAM: A Family of Large Action Models to Empower AI Agent Systems. **arXIv 2024**]

Agent Architectures

xLAM (eXtended LAM)

https://github.com/SalesforceAIResearch/xLAM

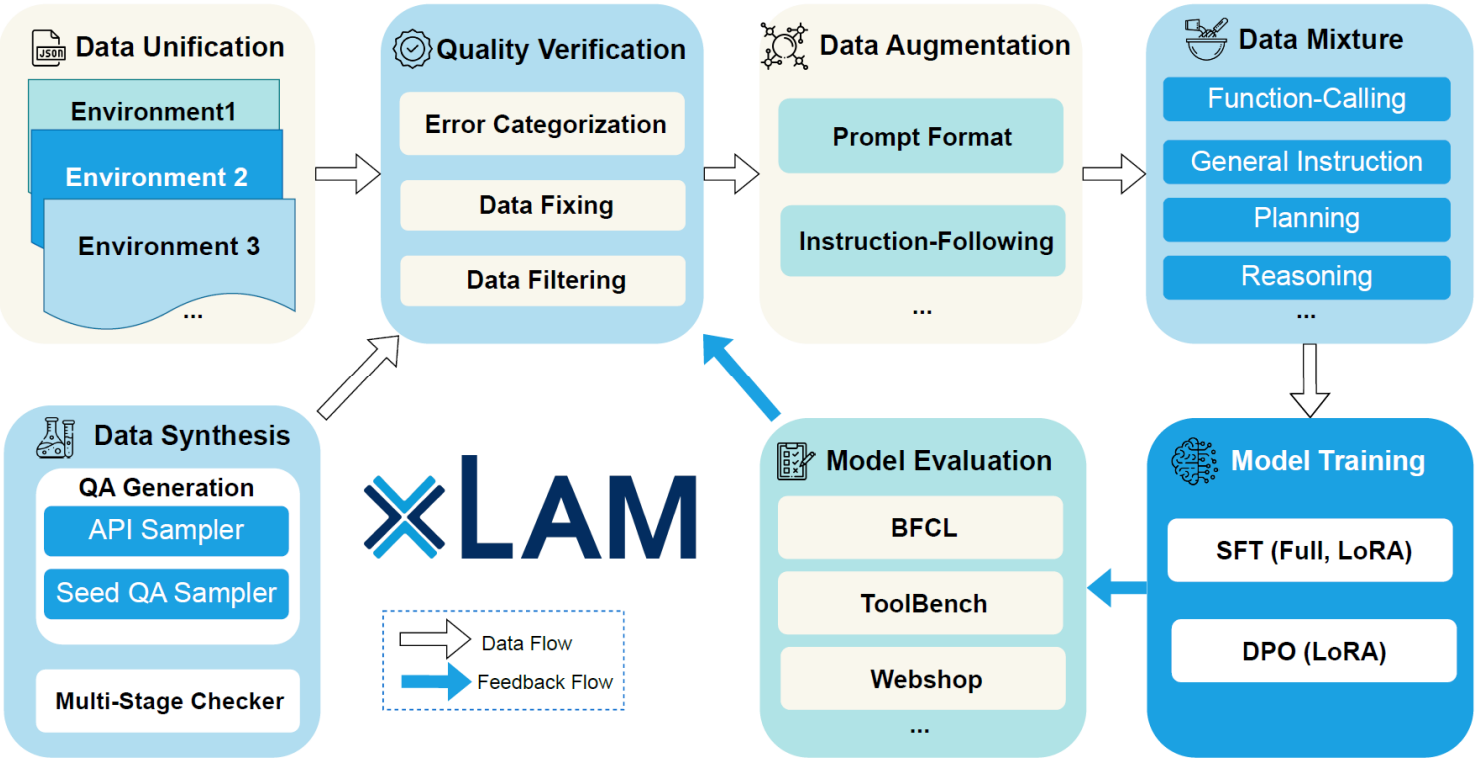


Figure 1: Overview of the data processing, training and evaluation of xLAM. We take the diagnostic feedback from the model evaluation results to iteratively improve the data quality

[Zhang el al. xLAM: A Family of Large Action Models to Empower AI Agent Systems. **arXiv 2024**]

- Data pipeline for training xLAM, including data unification, augmentation, quality verification, general instruction data synthesis, and preference data generation

[Data Unification] Unified data format in a function-calling style - **Function-calling formats** form the basis for how models understand and execute tasks

[Data Augmentation] ① Prompt Format Augmentation : Order Shuffling, Concatenation Tokens, ② Instruction-Following Augmentation : Task Instruction Rephrasing, Format Instruction-Following

[Quality Verification] Undefined Function Call, Incorrect Argument Type, Argument Hallucination, Low-Quality Reasoning and Planning

[Data Synthesis] APIGen; 3,673 APIs across 21 categories from ToolBench [8] to generate a total of 60,000 high-quality data using DeepSeek-V2-Chat and Mixtral-8x22B-Inst.

Agent Architectures

xLAM (eXtended LAM)

https://github.com/SalesforceAIResearch/xLAM

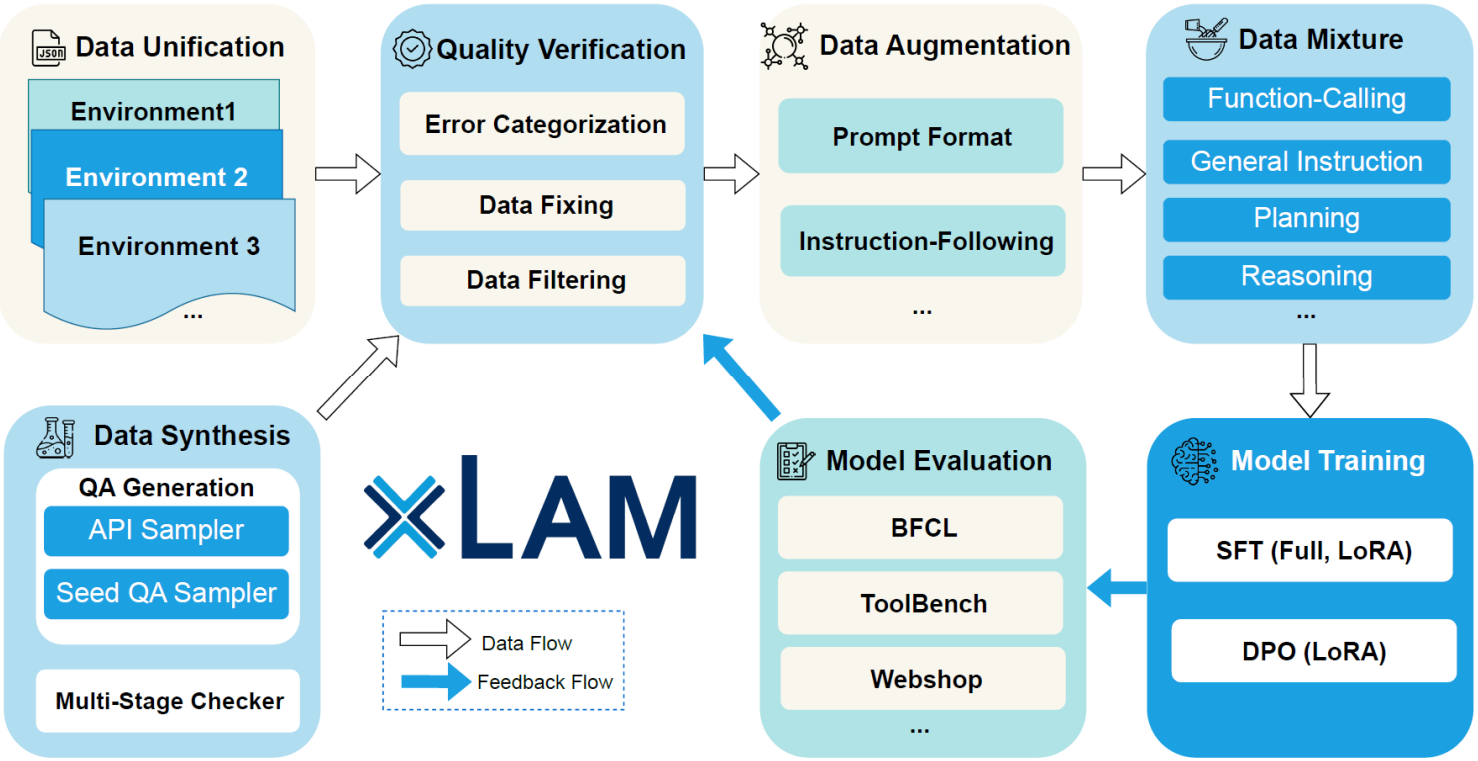


Figure 1: Overview of the data processing, training and evaluation of xLAM. We take the diagnostic feedback from the model evaluation results to iteratively improve the data quality

[Zhang el al. xLAM: A Family of Large Action Models to Empower AI Agent Systems. **arXiv 2024**]

[Data MIxture]

For **SFT** : Combine cleaned and augmented agent datasets, a synthetic function-calling dataset, and general instruction tuning datasets

For **DPO** : Prompt models to generate and rate responses for selected data from each source, then sample a subset for human verification. After adjustments to models and prompts, we classify the selected responses as rejected samples.

[Model Training] Use a supervised fine-tuning (SFT) approach, further aligning model checkpoints with the DPO method

For SFT, we use a full fine-tuning framework that employs the fully sharded data parallel algorithm. In the case of xLAM-8x22b-r, we integrate LoRA to better preserve the model's original capacities and prevent catastrophic forgetting. LoRA is also used for DPO alignment across all xLAM models.

Agent Architectures

Data Unification

- The **unified function calling format** consists of several modules: ***task instruction***, ***available tools***, ***format (task) instruction***, ***few-shot examples***, ***query***, and ***steps***.

- ***Tools*** define the agent's action space,
- ***format instruction*** specifies the output format the agent should follow when generating a response.
- In each **step**, the agent's output, the environment's feedback/execution results, and the user's follow-up input are organized into a dictionary.

✓ Compatible with various environments and tasks

✓ Make our data processing pipeline adaptable to different datasets and scalable to large amounts of data.

✓ Allow the modularized design for fine-grained data augmentation and quality verification, which are essential in improving agent data quality

```
{
  " unique_trajectory_id ": "id",
  " task_instruction ": " ... ",
  " few_shot_examples ": [],
  " query ": " The task or the question that the user provides .",
  " tools ": [
    {
      " name ": " api_name 1",
      " description ": " description of this api ",
      " parameters ": {
        " param 1": {
          " type ": " string ",
          " description ": "",
        },
      }
    },
  ],
  " steps ": [
    {
      " thought ": " thinking and /or planning process ",
      " tool_calls ": [
        {
          " name ": " api_name 1",
          " arguments ": {
            " argument 1": " xxx.",
            " argument 2": " xxx"
          }
        }
      ],
      " step_id ": 1,
      " next_observation ": " observations or feedbacks from the environment /
APIs after execution function ."
      " user_input ": " User follow up input at this turn if any ."
    },
  ],
}
```

Figure 4: Unified function calling data format

[Zhang el al. xLAM: A Family of Large Action Models to Empower AI Agent Systems. **arXIv 2024**]

Agent Architectures

xLAM (eXtended LAM)

https://github.com/SalesforceAIResearch/xLAM

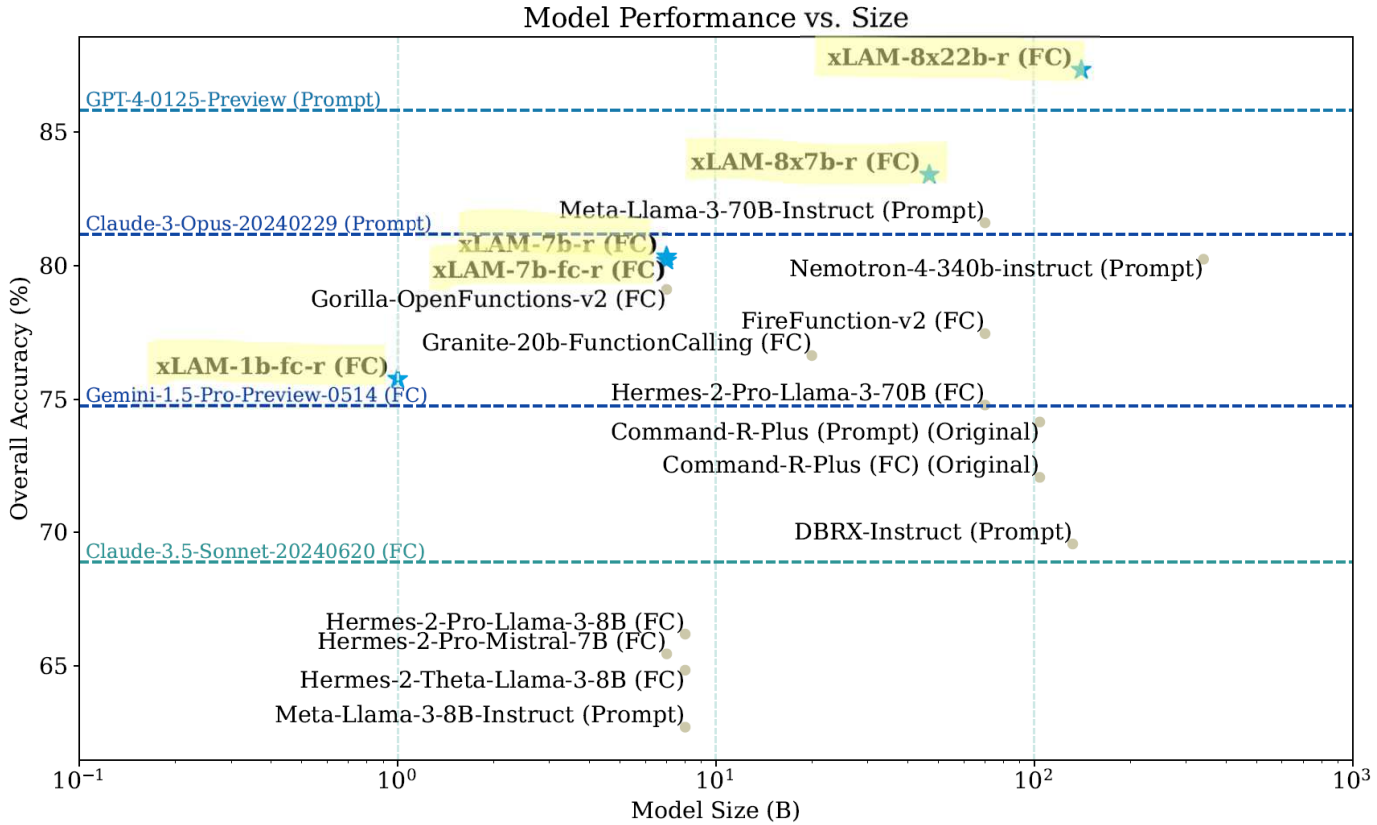


Figure 2: An overview of xLAM model performances on the Berkeley Function Calling Leaderboard v2 (cutoff date 09/03/2024). Our 8x22b model secures the top-1 position with wide margin on the leaderboard.

[Zhang el al. xLAM: A Family of Large Action Models to Empower AI Agent Systems. **arXIv 2024**]

- Several insights for agent model training

[Data Preprocessing] Highlight the importance of **data unification** and **augmentation** in enhancing dataset diversity and mitigating overfitting → Improve the generalizability of agent models across diverse environments

[Data Synthesis] the impact of scalable, high-quality data synthesis on agent model performance → Our synthetic dataset enabled xLAM models to achieve 4 of the top 20 positions on the Berkeley Function Calling Leaderboard, including securing the top-1 spot (Fig. 2), with smaller models achieving performance comparable to much larger counterparts, showing great potential in this direction.

Agent Architectures

Example prompt and output for function-calling using xLAM

[BEGIN OF TASK INSTRUCTION]
Based on the previous context and API request history, generate an API request or a response as an AI assistant.
[END OF TASK INSTRUCTION]

[BEGIN OF AVAILABLE TOOLS]
```
[
    {
        "name": "get_fire_info",
        "description": "Query the latest wildfire information",
        "parameters": {
            "location": {
                "type": "string",
                "description": "Location of the wildfire.",
                "required": true,
            },
            "radius": {
                "type": "number",
                "description": "The radius (in miles) around the location.",
                "required": false,
            }
        },
    },...
]
```
[END OF AVAILABLE TOOLS]

[BEGIN OF FORMAT INSTRUCTION]
Your output should be in the JSON format, which specifies a list of function calls. The example format is as follows. Please make sure the parameter type is correct. If no function call is needed, please make tool_calls an empty list "[]".
```
"""
{"thought": "the thought process, or an empty string", "tool_calls": [{"name": "api_name1", "arguments": {"argument1": "value1", "argument2": "value2"}}]}
"""
```
[END OF FORMAT INSTRUCTION]

[BEGIN OF QUERY]
Can you give me the latest information on the wildfires occurring in California?
[END OF QUERY]

[BEGIN OF HISTORY STEPS]
```
[
    {
        "thought": "Sure, what is the radius (in miles) around the location of the wildfire?",
        "tool_calls": [],  "step_id": 1,
        "next_observation": "",  "user_input": "User: Let me think... 50 miles."
    },
]
```
[END OF HISTORY STEPS]

Output:
```
{"thought": "", "tool_calls": [{"name": "get_fire_info", "arguments": {"location": "California", "radius": 50}}]}
```

[Zhang el al. xLAM: A Family of Large Action Models to Empower AI Agent Systems. **arXIv 2024**]

Juan Carlos Niebles, Salesforce

AgentOhana

https://github.com/SalesforceAIResearch/xLAM

- AgentOhana : Comprehensive **agent data collection** and **training pipeline**

- **Transform disparate data into a uniform format**, achieving seamless integration across multiple sources, and ensure high-quality trajectories by a **meticulous filtering process**
- Leveraging the **data standardization and unification**, our training pipeline preserves independent randomness across devices during both dataset partitioning and model training, thus avoiding the inadvertent introduction of biases into the training process.
- AgentOhana **unifies trajectories** across environments but also enhances **the overall quality and the reliability of the collected data**, as well as **the performance and the robustness of the model**.

- [**Solution to Data Heterogeneity**] Address the complex challenges associated with the consolidation of heterogeneous data sources pertaining to **multi-turn LLM agent trajectories**.
- [**Extensive Environmental Coverage**] Distinguishes itself by incorporating agent data from 10 distinct environments
- [**Standardization and Unification of LLM Agent data**] *Consistent format*. Enable the creation of a generic data loader, optimizing the dataset for agent training that maintains equilibrium across different data sources and preserves independent randomness across devices
- [**Large Agent Model**] Develop **XLAM-v0.1**, a robust large action model tailored for AI agents in AgentOhana

[Zhang el al. The AgentOhana: Designing Unified Data and Training Pipeline for Effective Agent Learning. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

AgentOhana: Unified Diverse Training Data

https://github.com/SalesforceAIResearch/xLAM

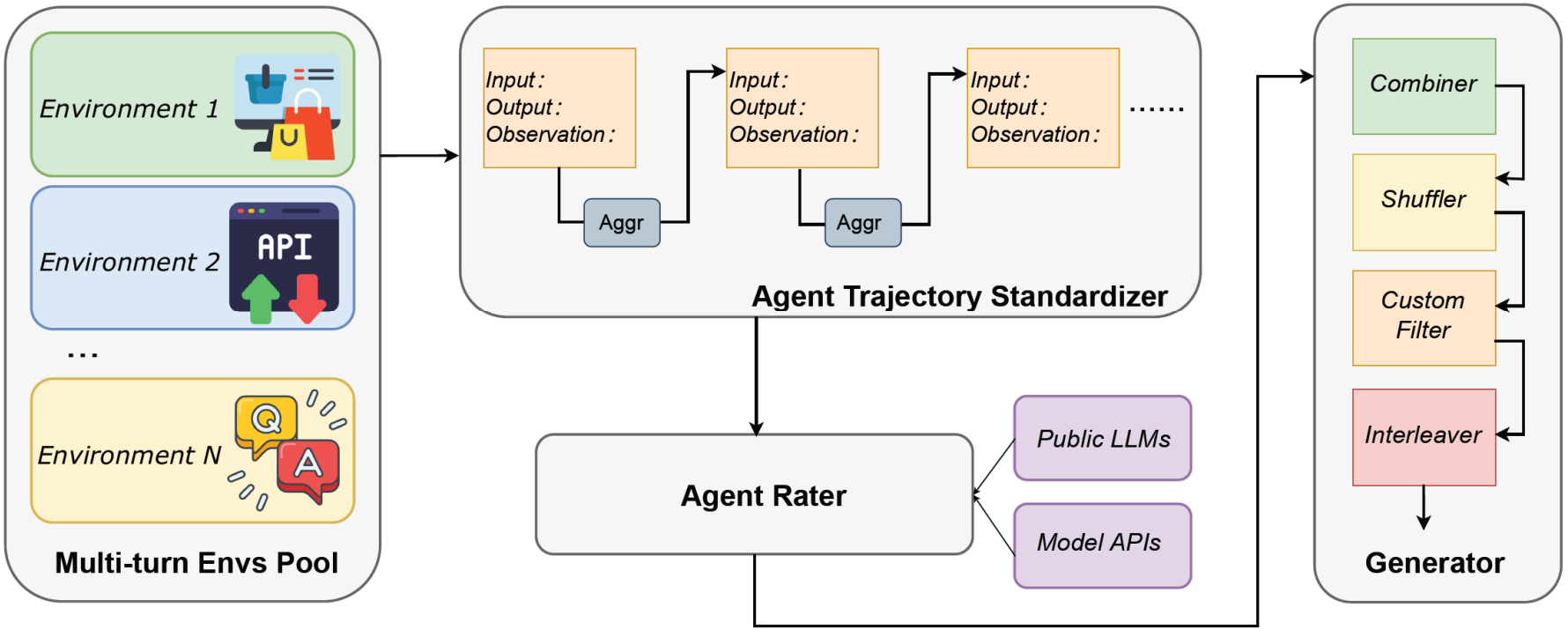


Figure 1: Workflow of ***AgentOhana***. A **homogeneous multi-turn data format** is designed to consolidate ***heterogeneous trajectories*** from diverse data sources. ***AgentRater*** then assesses and filters agent trajectories. Finally, a streaming data loader enables integration of various datasets and feeds data into a distributed training process at random.

[Zhang el al. The AgentOhana: Designing Unified Data and Training Pipeline for Effective Agent Learning. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

Agent Trajectory Data Example

https://github.com/SalesforceAIResearch/xLAM

1) Heterogeneity of Various Datasets

- Agent data formats vary significantly across different environments

- **HotPotQA**
- Consolidates the whole target trajectory into a single string under the ***prompt key***.
- Requires to retrieve ***User Query***, ***Thought***, ***Model Action***: i along with Env Observation: i for each step i th $\in$ [1,N] from a single string.

- **ToolAlpaca**
- Requires the identification and matching of *prompt inputs*, *model outputs*, and *observations* at each step, followed by the accurate aggregation of trajectory history prior to proceeding to the next step.

HotpotQA

```
{
  "question": "Which magazine was started first Arthur's Magazine or First for Women?",
  "answer": "Arthur's Magazine",
  "correct": true,
  "reward": 1.0,
  "halted": false,
  "error": false,
  "prompt": "
    Question: Which magazine was started first Arthur's Magazine or First for Women? Follow Action[param] format.\n Thought: I need to Search[Arthur's Magazine] and then Search[First for Women] to collect their starting information.\n Model Action 1: Search[Arthur's Magazine] \n Env Observation 1: Arthur's Magazine was published in 1844. \n Model Action 2: Search[First for Women]\n Env Observation 2: First for Women is a woman's magazine published in 1989. \n Model Action 3: Finish[Arthur's Magazine]\n Env Observation 3: Answer is CORRECT"
}
```

ToolAlpaca

```
[[
    "Begin!\n\nUSER: Can you send a GET request to link 1 and then retrieve the cookie by sending another GET request to link 2\n Follow Action: name \n Action Input:{param} format.",
    "ASSISTANT Thought: I need to send a request to set the cookie. \nASSISTANT Action: sendHttpRequest\nASSISTANT Action Input: {link1}\nASSISTANT Observation: ",
    "Status Code: 200.... Response 1\n",
    "ASSISTANT Thought: The request was successful.\nASSISTANT Action: getClientRequestData\nASSISTANT Action Input: {link 2}\nASSISTANT Observation: ",
    "Status Code: 200. ...\nASSISTANT Thought: ",
    "The cookie data was retrieved.\nASSISTANT Response: The request was successful."
  ],
  [
    false, true, false, true, false, true
  ]
]]
```

AgentOhana Trajectories

[Zhang el al. The AgentOhana: Designing Unified Data and Training Pipeline for Effective Agent Learning. **ICLR 2024** Workshop on LLM Agents]

Agent Architectures

Juan Carlos Niebles, Salesforce

AgentOhana Trajectories

2) Homogeneous Multi-Turn Agent Trajectory Standardization

- **Unified trajectory data format**

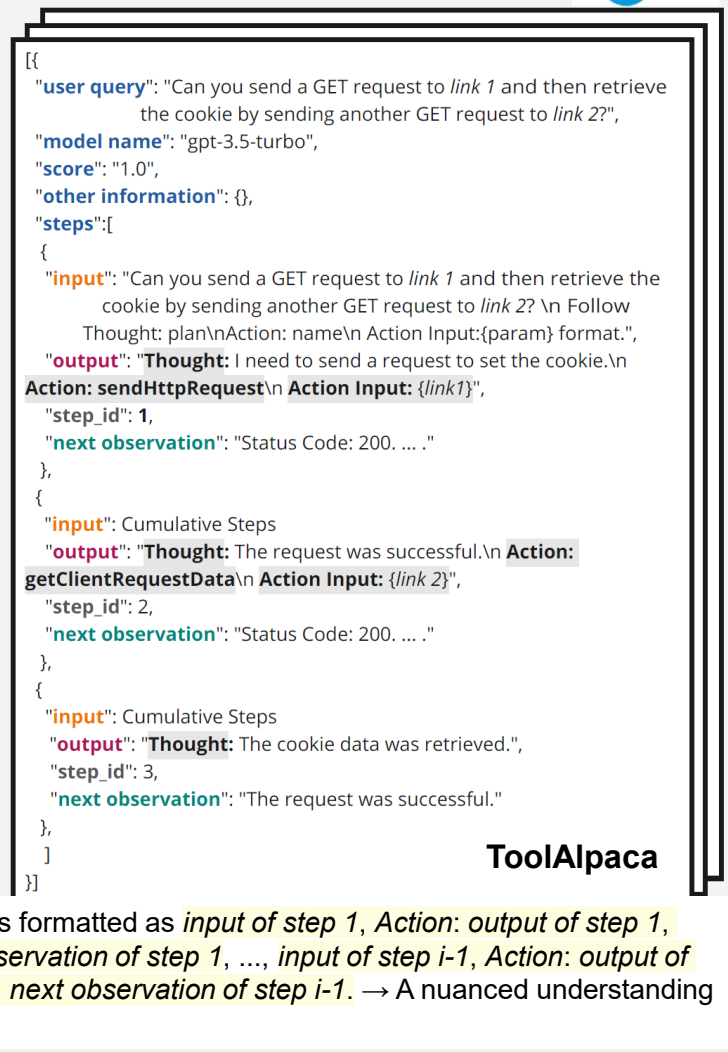
- Homogeneous JSON dictionary format to encapsulate all relevant content of each trajectory.;
- Incorporates all important elements such as ***user query*** to store the initial user query, ***model name*** to identify the corresponding model and ***score*** to log the available model performance score.
- Structured definition of a step for each interaction turn

✓ **Input** : Consolidates the current prompt and a historical record of past interactions

✓ **Output** : Captures the model's predictions, detailing its decision-making and planning.

✓ **Next observation** : Records environment's feedback, essential for the feedback loop and system adaption.





At *i*th step, the input is formatted as *input of step 1*, *Action*: *output of step 1*, *Observation*: *next observation of step 1*, ..., *input of step i-1*, *Action*: *output of step i-1*, *Observation*: *next observation of step i-1*. → A nuanced understanding of the trajectory.

[Zhang el al. The AgentOhana: Designing Unified Data and Training Pipeline for Effective Agent Learning. **ICLR 2024** Workshop on LLM Agents]

AgentOhana Trajectories

3) AgentRater

- **Agent trajectories** represent a complex subset of data distinct from general and straightforward instructional data
- The challenge with agent trajectories extends to the evaluation of performance and quality.
- *A high reward does not necessarily indicate a flawless trajectory*. For example, an agent might generate invalid actions during intermediate steps of a task.
- Design **AgentRater** to **rate the agent trajectory** based on strong public models such as Mistral (Jiang et al., 2023) or close-world APIs such as ChatGPT (OpenAI, 2023). - Rate the whole trajectory on agent data.

[BEGIN OF JSON DICT FILE]
{**Given Agent Trajectory**}
[END OF JSON DICT FILE]

Given above json dict contains a trajectory, please rate according to the **overall accuracy and efficiency** of the model output to the input and observation.
Here are **keys** for the json dict file:
Initial Input of Step 1 ###: this indicates the initial input of the first step
Model Output of Step 1 ###: this indicates the model output based on the Initial Input of the first step
Observation of Step 1 ###: this indicates the observation from environments based on the Model Output of the first step
Model Output of Step 2 ###: this indicates the model output based on the observation of the first step and the previous history
Observation of Step 2 ###: this indicates the observation from environments based on the Model Output of the second step
Model Output of Step 3 ###: this indicates the model output based on the observation of the second step and the previous history
So on and so forth for the rest of the steps.

Each assistant receives **a score on a scale of 0 to 5**, where a higher score indicates higher level of the overall model accuracy and efficiency.
Please provide your evaluation as follows:
1. A single line containing **a numerical score** indicating the evaluation.
2. In the subsequent line, please provide **a detailed explanation** supporting your score, **focusing on the criteria** of accuracy and efficiency. Ensure your evaluation avoids any potential bias.

Figure 3: A prompt template for the AgentRater, where an open-source model (e.g., Mistral) or close-world API (e.g., ChatGPT) will rate the whole agent trajectory based on criterias and then assign a score from 0 - 5

[Zhang el al. The AgentOhana: Designing Unified Data and Training Pipeline for Effective Agent Learning. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

AgentOhana Trajectories

4) Generic Dataloader

- Serves as a central component, facilitating seamless integration of diverse datasets into the training process

4.1) AgentModelDatasetBase class

- Streamline common tasks such as prompt formatting while providing a virtual template for creating individual datasets.

❖ Intricate Issues;

- ✓ Machine-assisted filter
- ✓ Users may prefer a certain level of control over data quality.
- ✓ Randomness of data batching from different dataset

```
class WebshopMultiTurn(AgentModelDatasetBase):

    # we can further filter out trajectories at this stage
    @staticmethod
    def _high_score_filter_generator(data, score=0.8):
        for d in data:
            if d["score"] >= score:
                yield {"prompt": d["input"], "chosen": d["output"]}

    def create_datasets(self, seed=None):
        train_data = load_dataset(
            ...
            streaming = self.args.streaming,
        )
        train_data = IterableDataset.from_generator(
            self._high_score_filter_generator,
            gen_kwargs={"data": train_data}
        )
        train_data = train_data.shuffle(seed=seed, buffer_size=1000)
        return train_data
```

[Zhang el al. The AgentOhana: Designing Unified Data and Training Pipeline for Effective Agent Learning. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

AgentOhana Trajectories

4.2) Custom Dataset Creation

- **Individual dataset**

① Load individual raw data prepared typically via the streaming mode.

② For each dataset, we can optionally introduce the filter generator to further customize the selection of data just before feeding it into the trainer. (Ex; Data with relatively low scores will be further evaluated and removed.)

③ Shuffle this dataset randomly with controlled seeding

```
toolbench_multi_turn = ToolBenchMultiTurn(tokenizer, script_args)
webshop_multi_turn = WebshopMultiTurn(tokenizer, script_args)
...

data = [toolbench_multi_turn, webshop_multi_turn, ...]

sample_probs = [0.1, 0.1, ...]

# a device-dependent seeding will be utilized based on the combination of
# the given default seed and the process ID
seed = init_device_seed(seed=42)

train_dataset, eval_dataset = \
    interleave_data(
        data_objects=data,
        sample_probs=sample_probs,
        seed=seed)
```

- **Combined dataset**; Ensure randomness for multiple datasets

- Employ the ***init device seed function*** to diversify the controlled seeds based on the process ID when data parallelism is utilized across multiple devices.
- Aim to maintain a balanced distribution of data by *partitioning*, *shuffling* and *interleaving* data across devices while preserving *randomness*, thus enhancing the *robustness* and *reproducibility* of our training procedure.

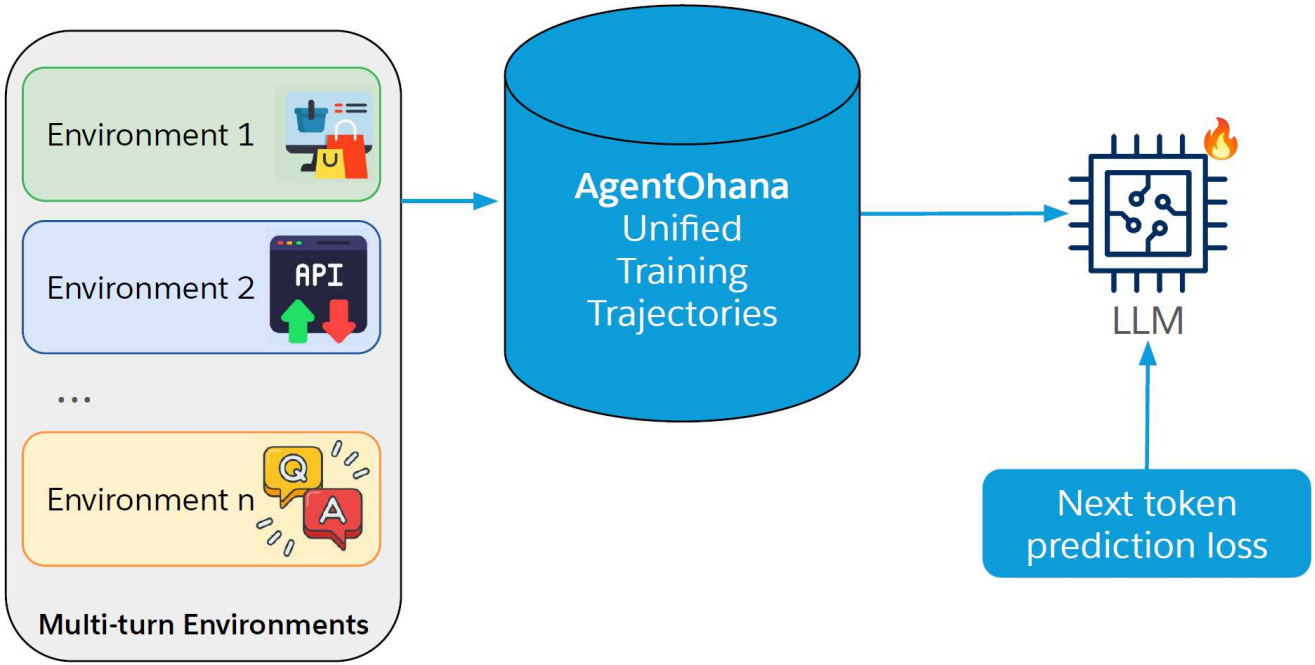
[Zhang el al. The AgentOhana: Designing Unified Data and Training Pipeline for Effective Agent Learning. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

Training xLAM (Large Action Model)

https://github.com/SalesforceAIResearch/xLAM

Supervised Fine-tuning (SFT)



- Adopt a supervised fine-tuning approach
- to enhance the performance of agent model, **xLAM-v0.1**, which was initially pre-trained on the **Mixtral-8x7B-Instruct-v0.1** model.
- To execute this fine-tuning process, we capitalized on the capabilities of **AgentOhana**. Our fine-tuning procedure was conducted concurrently on 8 Nvidia H100 GPUs, utilizing the **QLoRA framework**.

- Our **dataset compilation strategy**;
- ensured the creation of a comprehensive training corpus with a balanced distribution of data from various sources.
- Preserve independent randomness, prevent the introduction of unintended biases

- Multi-epoch training;
- Throughout the fine-tuning process, our model traversed each individual dataset approximately 3 times on average.

[Zhang el al. The AgentOhana: Designing Unified Data and Training Pipeline for Effective Agent Learning. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

xLAM: Results

https://github.com/SalesforceAIResearch/xLAM

LLM	LAA Architecture					
	ZS	ZST	ReAct	PlanAct	PlanReAct	BOLAA
Llama-2-70b-chat (Touvron et al., 2023)	0.0089	0.0102	0.4273	0.2809	0.3966	0.4986
Vicuna-33b (Zheng et al., 2023b)	0.1527	0.2122	0.1971	0.3766	0.4032	0.5618
Mixtral-8x7B-Instruct-v0.1 (Jiang et al., 2024)	0.4634	0.4592	0.5638	0.4738	0.3339	0.5342
GPT-3.5-Turbo	0.4851	0.5058	0.5047	0.4930	0.5436	0.6354
GPT-3.5-Turbo-Instruct	0.3785	0.4195	0.4377	0.3604	0.4851	0.5811
GPT-4-0613	0.5002	0.4783	0.4616	**0.7950**	0.4635	0.6129
xLAM-v0.1	**0.5201**	**0.5268**	**0.6486**	0.6573	**0.6611**	**0.6556**

Table 2: Average reward on the WebShop environment.

LLM	LAA Architecture				
	ZS	ZST	ReAct	PlanAct	PlanReAct
Mixtral-8x7B-Instruct-v0.1 (Jiang et al., 2024)	0.3912	0.3971	0.3714	0.3195	0.3039
GPT-3.5-Turbo	0.4196	0.3937	0.3868	0.4182	0.3960
GPT-4-0613	**0.5801**	**0.5709**	**0.6129**	**0.5778**	**0.5716**
xLAM-v0.1	0.5492	0.4776	0.5020	0.5583	0.5030

Table 3: Average reward on the HotpotQA environment.

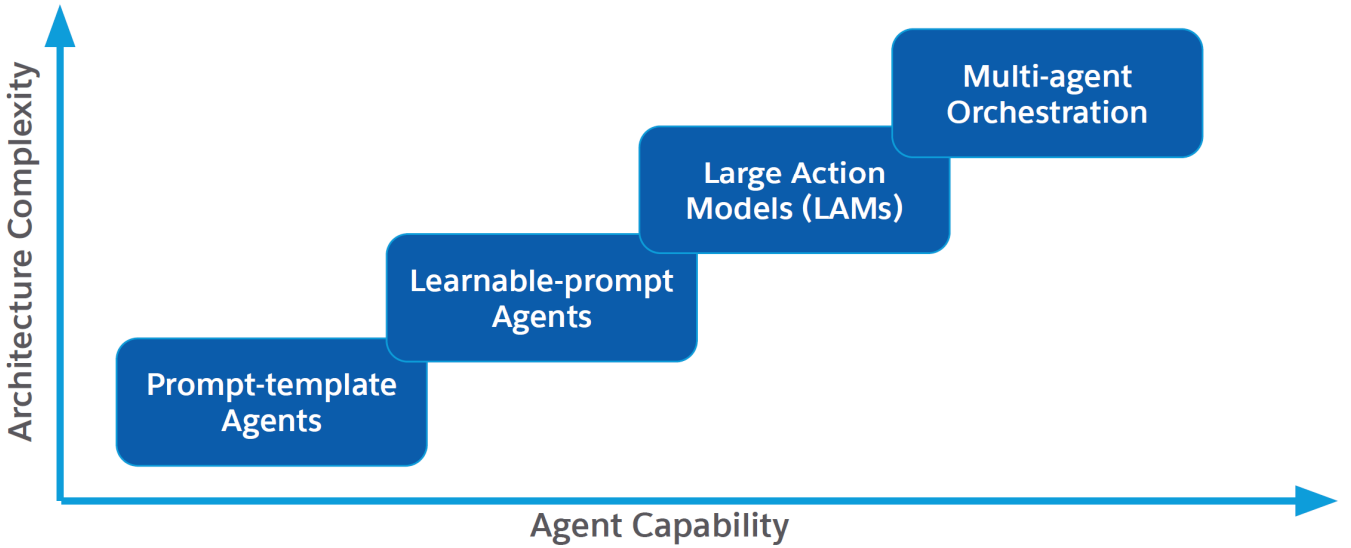
[Zhang el al. The AgentOhana: Designing Unified Data and Training Pipeline for Effective Agent Learning. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

Language-based AI Agents

➢ **Multi-agent Orchestration**

- 여러 개의 에이전트(AI 모델 또는 시스템)가 협력하여 복잡한 작업을 수행하도록 조정하는 방법
- 각 에이전트는 특정 역할이나 기능을 맡고 있으며, **orchestrator**는 이들 간의 **작업 분담**, **통신**, **순서 제어** 등을 맡아 목표를 달성하게 만듦. 이는 단일 강력한 에이전트 하나로 모든 것을 처리하는 대신, 여러 특화된 에이전트를 조합하여 **모듈화**와 **확장성**을 얻는 전략임



- Architecture Complexity : High
  - 다수의 에이전트들이 계층적(Hierarchical), 병렬적(Parallel), 조건적(Conditional) 흐름에 따라 상호작용하고, 동적 라우팅과 상태 관리가 요구됨.
- Agent Capability : High
  - 복합적인 reasoning, 멀티모달 처리, 상호대화 능력 등을 가진 에이전트

Juan Carlos Niebles, Salesforce

From Single Agents to Multiple Agents

https://github.com/SalesforceAIResearch/AgentLite

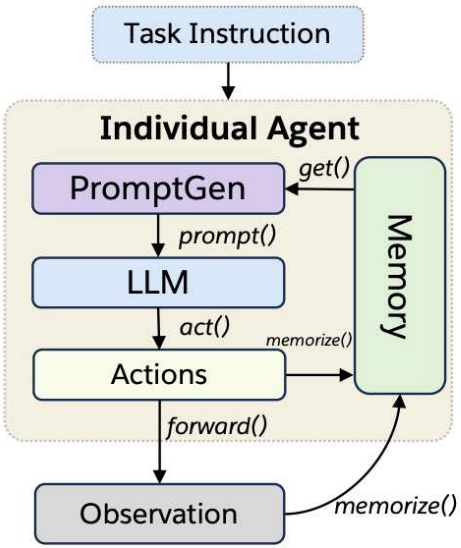
AgentLite

release v0.1.2 | Python 3.9+ | pypi v0.1.12 | License Apache | Stars 593

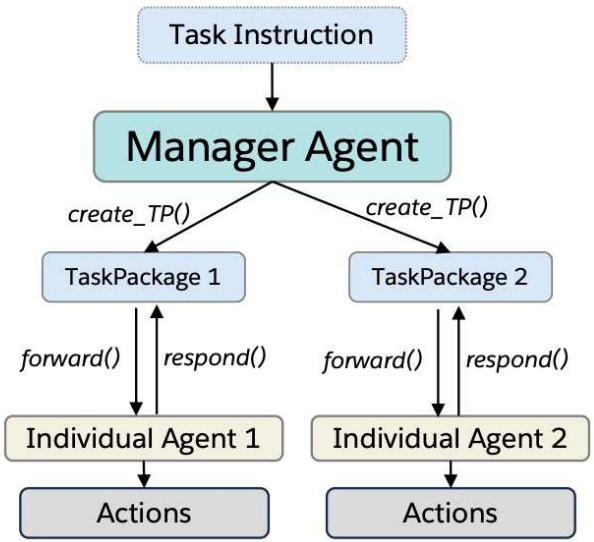
AgentLite is a research-oriented library designed for ***building and advancing LLM-based task-oriented agent systems***. It simplifies the implementation of new agent/multi-agent architectures, enabling easy orchestration of multiple agents through a manager agent

Key Features

- **Lightweight Codebase**: Designed for easy implementation of new Agent/Multi-Agent architectures.
- **Task-oriented LLM-based Agents**: Focus on building agents for specific tasks, enhancing their performance and capabilities.
- **Research-oriented Design**: A perfect tool for exploring advanced concepts in LLM-based multi-agent systems.



(a) Individual Agent



(b) Manager Agent

[Liu. **AgentLite**: A Lightweight Library for Building and Advancing Task-Oriented LLM Agent System. 2024]

Juan Carlos Niebles, Salesforce

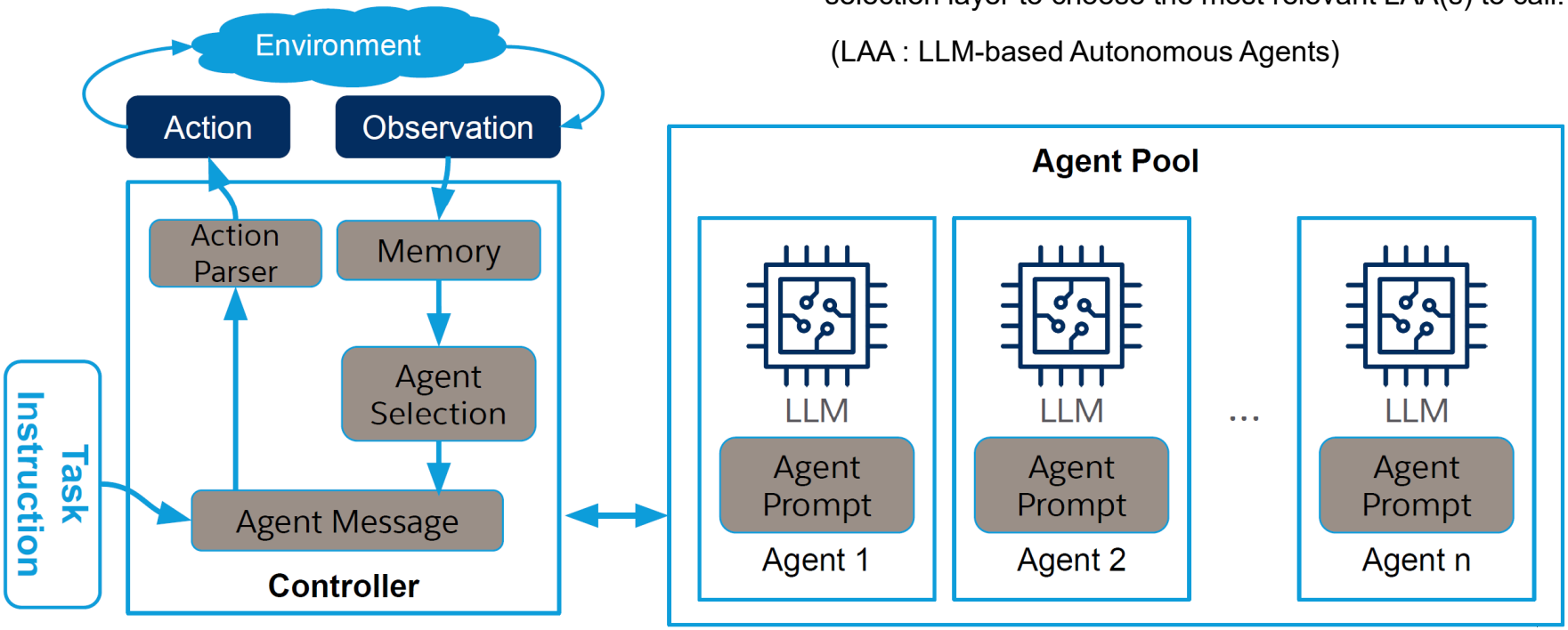
BOLAA: Orchestrating multiple agents

https://github.com/salesforce/BOLAA

A strategy to **orchestrate multiple LAAs** such that each labor **LAA** focuses on one type of action, i.e. **BOLAA**, where a controller manages the communication among multiple agents.

- **Labor agents pool** : Manages multiple LAAs (Ex; *Click* LLA, *Search* LLA)
- **Controller** : Selectively call LAAs from agents pool – Has agents selection layer to choose the most relevant LAA(s) to call.

(LAA : LLM-based Autonomous Agents)



[Zhiwei Liu. “BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents”. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

BOLAA: Orchestrating multiple agents

https://github.com/salesforce/BOLAA

- **Agent Selection;** One the core parts for orchestration
  - **Heuristic-based method** : Pre-define rules for selecting the labor LAA. Rules could be defined based on observation, generated actions, etc.
  - **LLM-based method** : Designing the controller based an LLM, and enabling the labor agent selection as an action generation process of the LLM
  - **The *controller* is *functioning* as the *orchestrator agent***, and **its *action* is to *select the optimal labor agent***.

After obtaining the response from the labor LAA, the controller parses it to an executable action and then interacts with the environment.

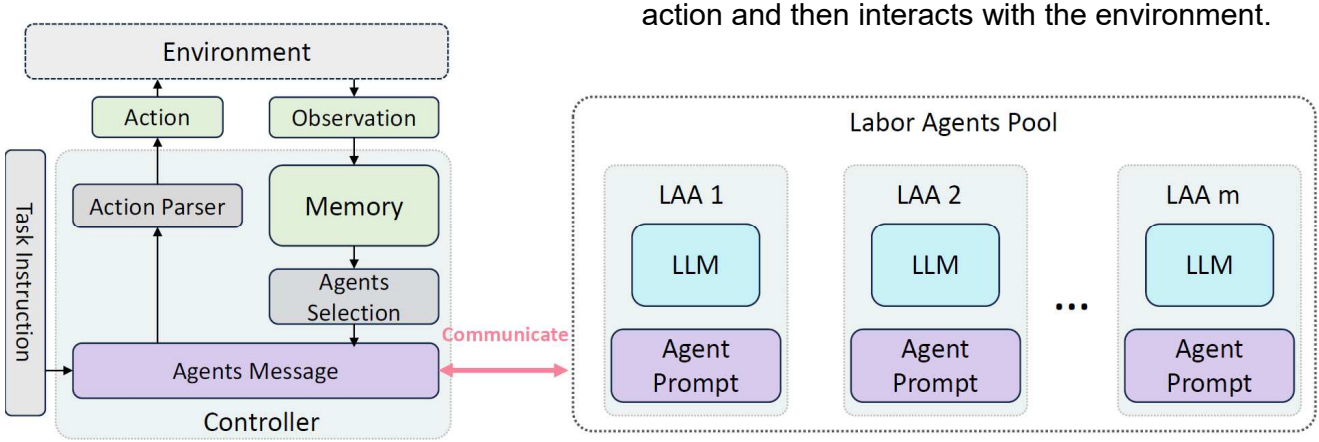


Figure 3: The BOLAA architecture, which employs a controller to orchestrate multiple LAAs.

- **Communication;** Other core parts for orchestration
  - After selecting the labor LAA, the controller constructs the message for the selected LAA and builds the communication.
  - **Heuristic-based method** : Generates the communication message by a pre-defined template, which ingests the observation from labor agents as a message for controller to send out.
  - **LLM-based method** : Labor agents' response are organized by a prompt template. Then, LLM is prompted to generate communication messages to those labor agents.
  - ✓ The prompt for LLM includes *the labor agent information*, *the task instructions*, *history executions* and *demonstration examples*.

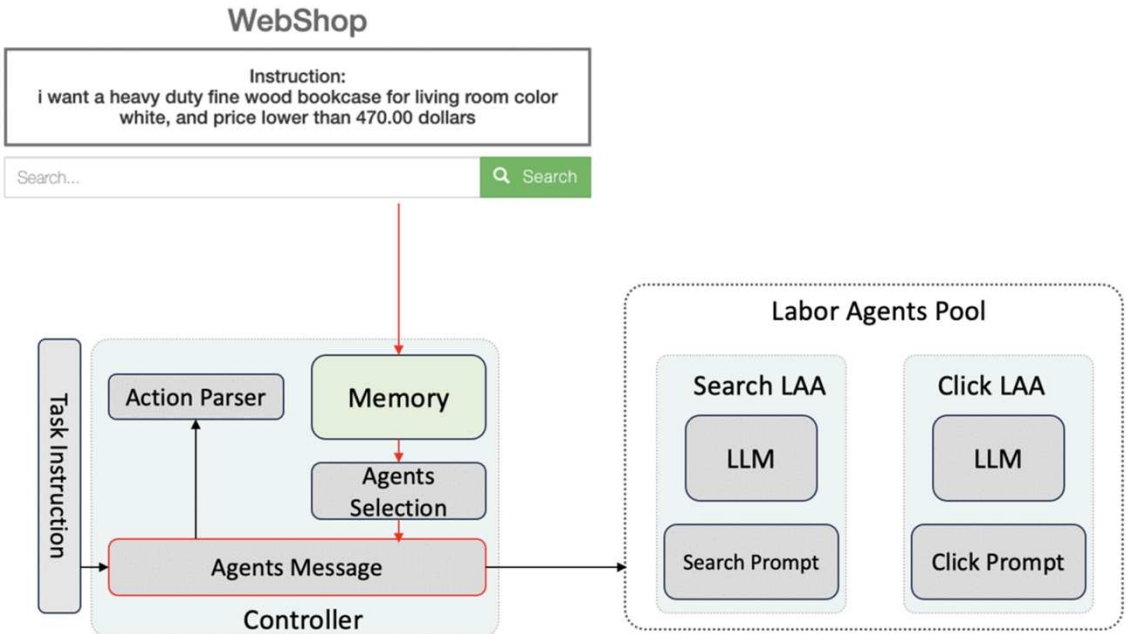
[Zhiwei Liu. "BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents". **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

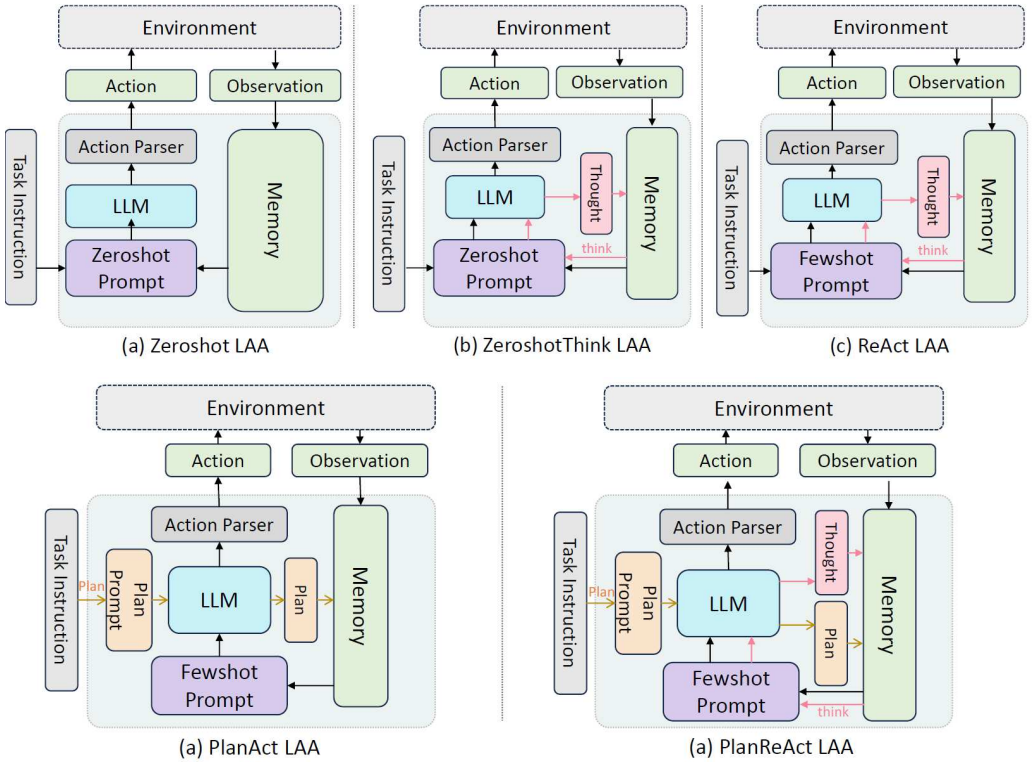
BOLAA: Orchestrating multiple agents

https://github.com/salesforce/BOLAA

- We tested on two types of enviroments: the webshop navigation environment, and HotPotQA enviroment.
- An example of the BOLAA web agent simulation on webshop enviroment is:



- Besides BOLAA architecture, we devise five standard LAA architectures, the Zeroshot (ZS), Zeroshot-Think (ZST), ReAct, PlanAct, PlanReAct as follows:



[Zhiwei Liu. “BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents”. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

BOLAA Multi-Agent Results

https://github.com/salesforce/BOLAA

Table 1: Average reward in the **WebShop** environment. Len denotes the maximum context length. Bold results denote the best results in one row, i.e. best LAA architecture w.r.t. one LLM. Underline results denote the best performance in one column, i.e. best LLM regarding one LAA architecture.

LLM	Len.	LAA Architecture					
		ZS	ZST	ReAct	PlanAct	PlanReAct	BOLAA
fastchat-t5-3b	2k	0.3971	0.2832	0.3098	0.3837	0.1507	**0.5169**
vicuna-7b	2k	0.0012	0.0002	**0.1033**	0.0555	0.0674	0.0604
vicuna-13b	2k	0.0340	0.0451	0.1509	0.3120	0.4127	**0.5350**
vicuna-33b	2k	0.1356	0.2049	0.1887	0.3692	0.3125	**0.5612**
llama-2-7b-chat	4k	0.0042	0.0068	0.1248	0.3156	0.2761	**0.4648**
llama-2-13b-chat	4k	0.0662	0.0420	0.2568	**0.4892**	0.4091	0.3716
llama-2-70b-chat	4k	0.0122	0.0080	0.4426	0.2979	0.3770	**0.5040**
mpt-7b-instruct	8k	0.0001	0.0001	0.0573	0.0656	**0.1574**	0.0632
mpt-30b-instruct	8k	0.1664	0.1255	0.3119	0.3060	0.3198	**0.4381**
xgen-8k-7b-instruct	8k	0.0001	0.0015	0.0685	0.1574	0.1004	**0.3697**
longchat-7b-16k	16k	0.0165	0.0171	0.069	0.0917	0.1322	**0.1964**
longchat-13b-16k	16k	0.0007	0.0007	0.2373	0.3978	**0.4019**	0.3205
text-davinci-003	4k	0.5292	0.5395	0.5474	0.4751	0.4912	**0.6341**
gpt-3.5-turbo	4k	0.5061	0.5057	0.5383	0.4667	0.5483	**0.6567**
gpt-3.5-turbo-16k	16k	0.5657	0.5642	0.4898	0.4565	0.5607	**0.6541**

Table 3: Average reward in the **HotPotQA** environment.

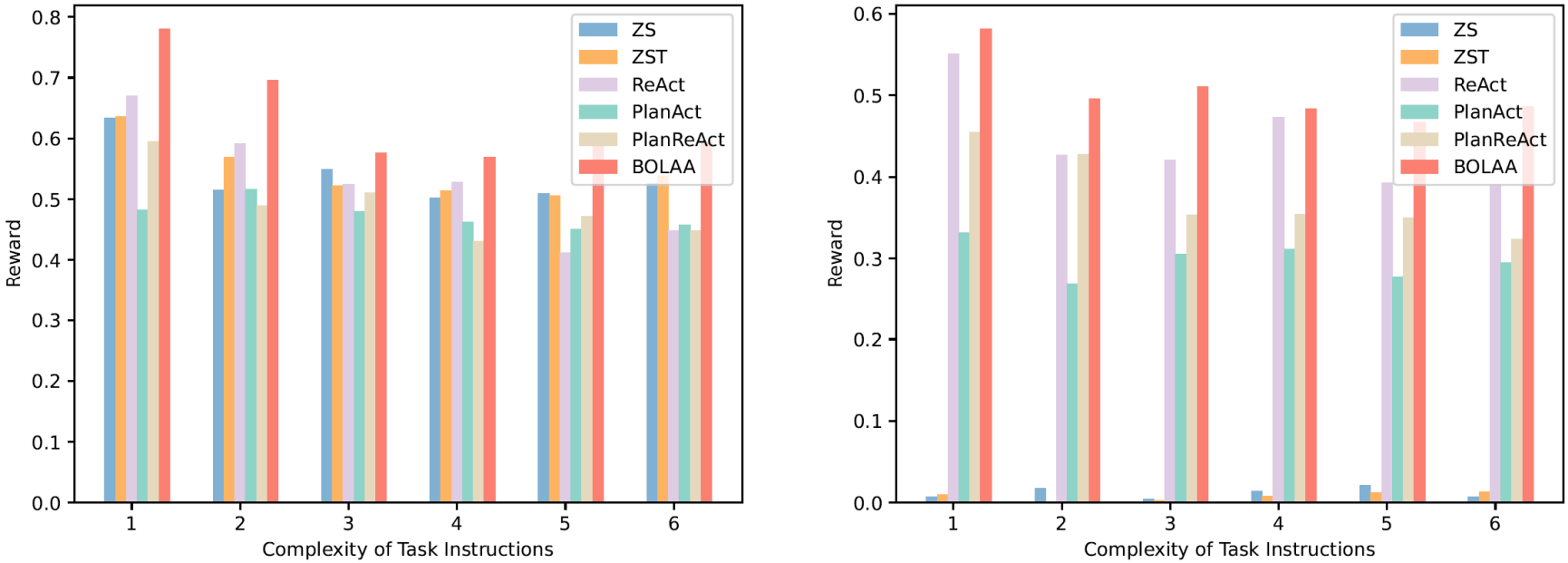
LLM	Len.	LAA Architecture					
		ZS	ZST	ReAct	PlanAct	PlanReAct	BOLAA
fastchat-t5-3b	2k	0.0252	0.0067	0.0692	0.1155	0.0834	**0.1221**
vicuna-7b	2k	0.1339	0.0797	0.0318	0.0868	0.0956	**0.1521**
vicuna-13b	2k	0.1541	0.0910	0.2637	0.1754	0.2075	**0.2721**
vicuna-33b	2k	0.2180	0.2223	0.2602	0.1333	0.2016	**0.2754**
llama-2-7b-chat	4k	0.0395	0.0207	**0.2624**	0.1780	0.1417	0.2613
llama-2-13b-chat	4k	0.1731	0.2313	0.2521	0.2192	0.2177	**0.2773**
llama-2-70b-chat	4k	0.2809	0.3207	0.3558	0.1424	0.1797	**0.3681**
mpt-7b-instruct	8k	0.0982	0.0483	0.1707	0.1147	0.1195	**0.1775**
mpt-30b-instruct	8k	0.1562	0.2141	0.3261	0.2224	0.2315	**0.3521**
xgen-8k-7b-instruct	8k	0.1502	0.1244	0.1937	0.1116	0.1096	**0.2231**
vicuna-7b-16k	16k	0.0773	0.1053	**0.2554**	0.1759	0.1642	0.2347
longchat-7b-16k	16k	0.0791	0.0672	**0.2161**	0.1296	0.0971	0.1917
longchat-13b-16k	16k	0.1083	0.0562	0.2387	0.1623	0.1349	**0.2433**
text-davinci-003	4k	0.3430	0.3304	0.4503	0.3577	0.4101	**0.4743**
gpt-3.5-turbo	4k	0.3340	0.3254	0.3226	0.2762	0.3192	**0.3541**
gpt-3.5-turbo-16k	16k	0.3027	0.2264	0.1859	0.2113	0.2251	**0.3225**

[Zhiwei Liu. “BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents”. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

BOLAA: Task Instruction Complexity

https://github.com/salesforce/BOLAA



(a) text-davinci-003

(b) Llama-2-70b

Figure 4: The reward w.r.t. task complexity in WebShop. Each bar represents one LAA.

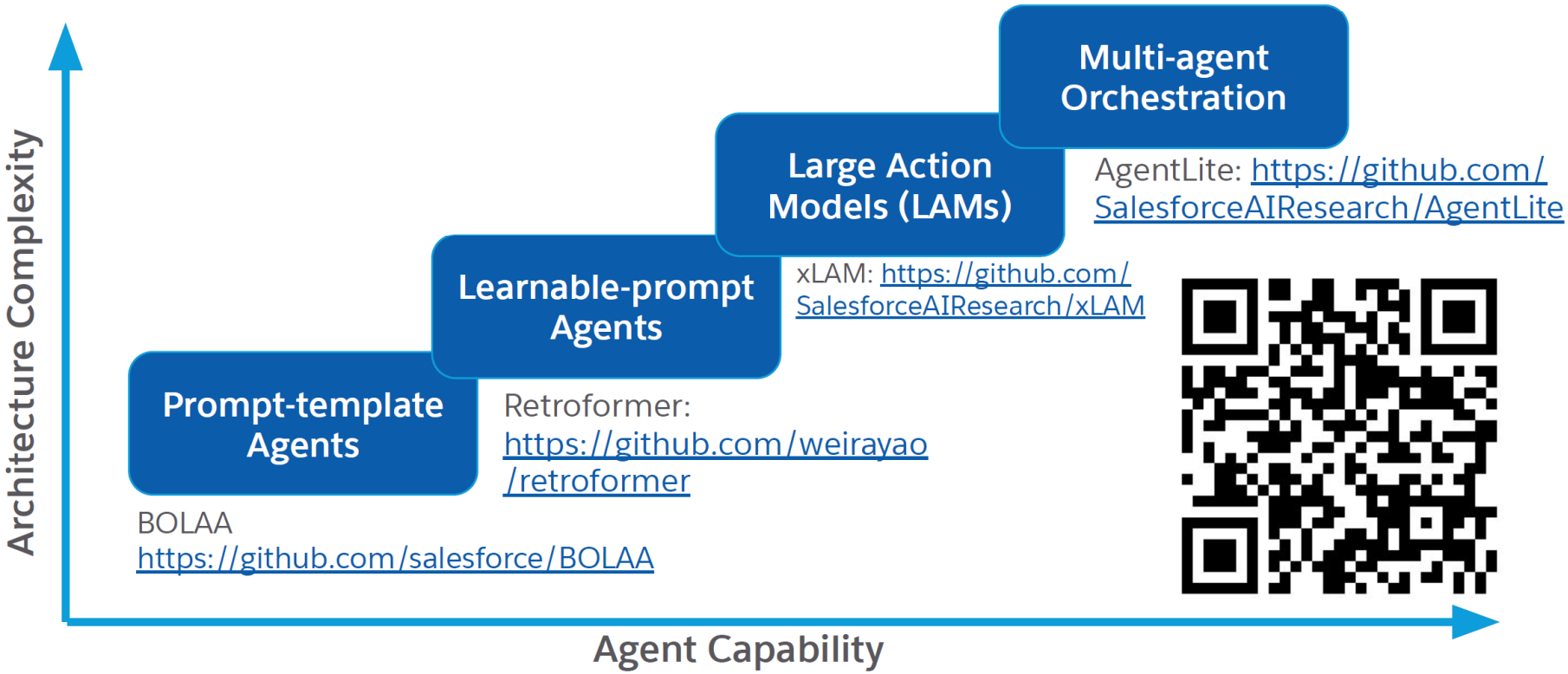
Task Complexity = number of attributes described in query

[Zhiwei Liu. “BOLAA: Benchmarking and orchestrating LLM-augmented autonomous agents”. **ICLR 2024** Workshop on LLM Agents]

Juan Carlos Niebles, Salesforce

Resources to Get Started

Salesforce AI Research Open Source Ecosystem



What's next?

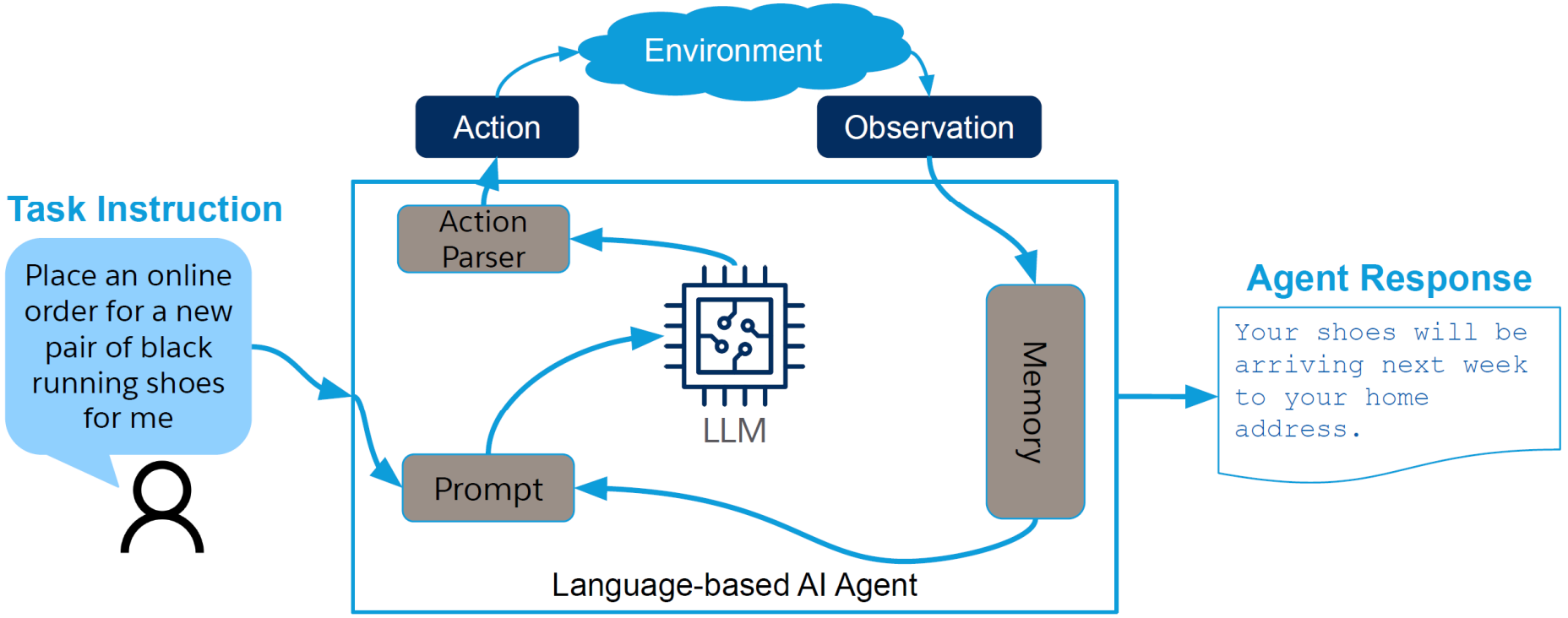
Interesting avenues to further explore



Juan Carlos Niebles, Salesforce

High-level Language-based Agent framework

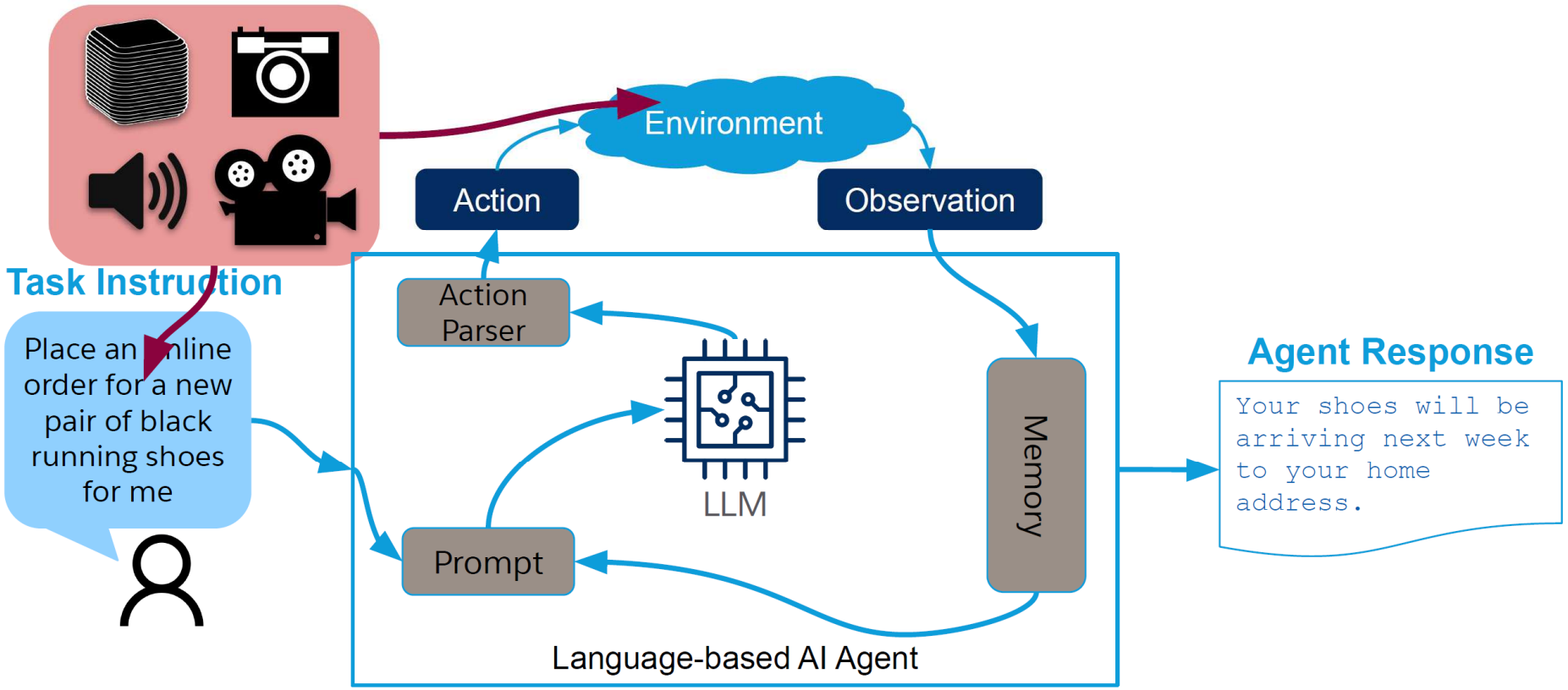
https://github.com/salesforce/BOLAA



Juan Carlos Niebles, Salesforce

Modality

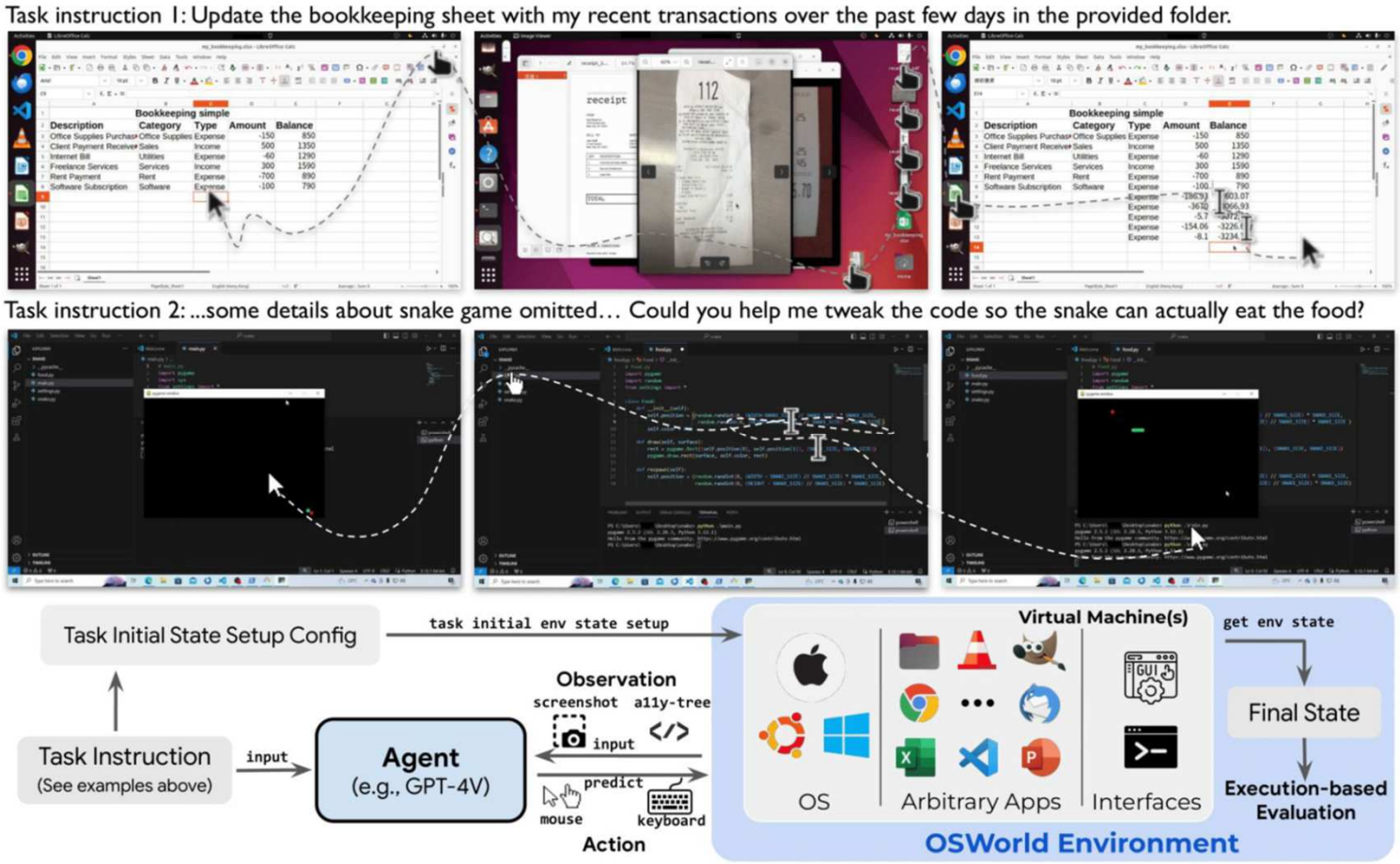
https://github.com/salesforce/BOLAA



Juan Carlos Niebles, Salesforce

OSWorld

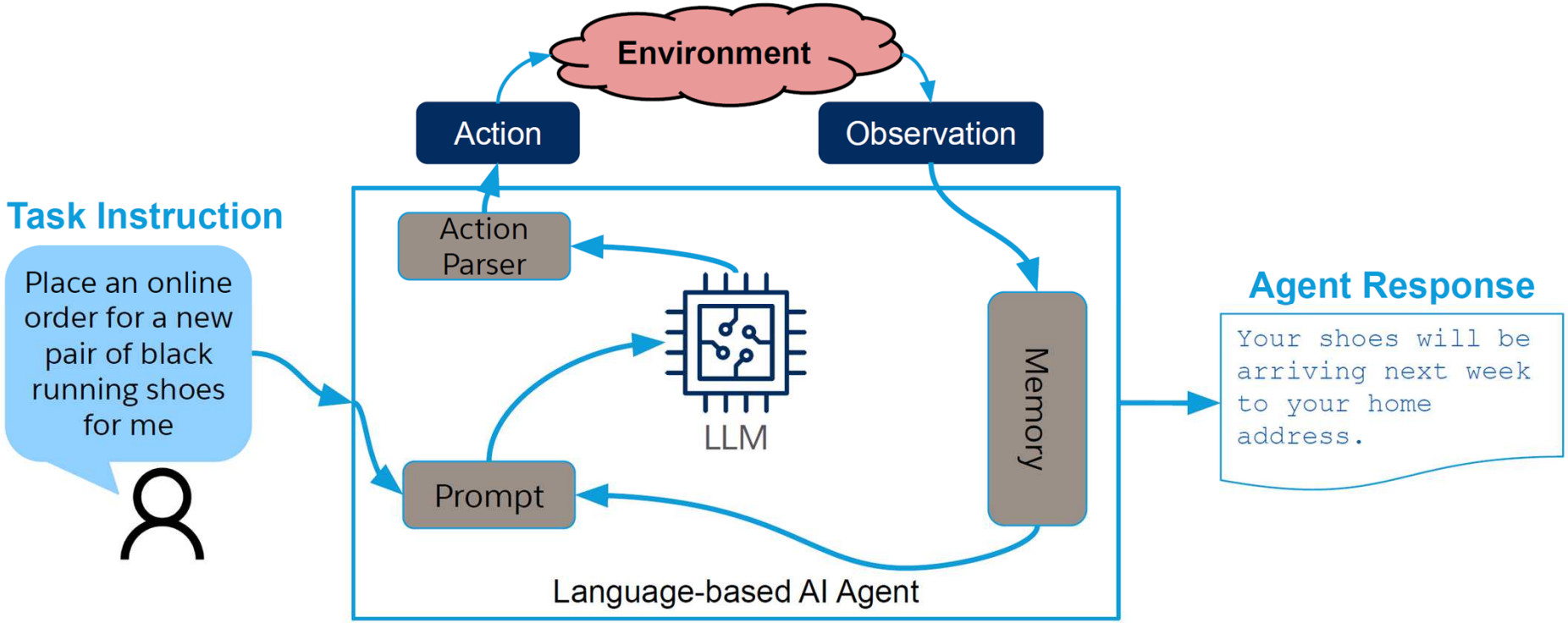
https://github.com/xlang-ai/OSWorld



[Xie et al. Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. 2024]

Benchmarking & Environments

Juan Carlos Niebles, Salesforce



Learning & Generalization

Juan Carlos Niebles, Salesforce

